研究報告
電子航法研究所報告
No.77 1993.9

ELECTRONIC NAVIGATION
RESEARCH INSTITUTE PAPERS
No.77 September 1993

エキスパートシステムESHELLによるコンフリクトフリー・プランニング・モデル

仁 科 光 雄*

Conflict-Free Planning Model for Air Traffic Control Using Expert System ESHELL

Mitsuo NISHINA

Abstract

Attwooll presented an idea of precise air traffic control by preassigning time slots to individual flights in the traffic (GBR, 1977). It was not practicable at that time. But now, using scheduling technique in expert system and high accurate navigation technique in FANS concept of ICAO, we have the feasibility of the idea.

As the first step to the Attwooll's method, CONFLICT-FREE PLANNING MOODEL was developed by using expert system ESHELL based on LISP.

Its function is to generate deconflicted ATC clearances of all domestic airlines' flights in Japan based on time table. As the result of test run consisted of 30 flights, the CPU time required 2 hours and 50 minutes on the running condition of 4 MIPS main frame computer and LISP interpreter use.

The full size model of about 1,200 domestic flights will run in ten minute order, if high speed workstation and high performance expert system are applied.

* 電子航法評価部

1．はじめに

わが国の航空管制におけるフローコントロールの必要性については，別稿「離散型シミュレーション言語GPSSによる航空交通流モデル」に述べた[1]。

フローコントロールとして，具体的には出発規制の方法が検討されている。これは諸外国でも例があり，当初この方法から入ることは妥当であろう。

同じフローコントロールでも，この出発規制と対照的に，航空管制に精密スケジューリングの方法を積極的に導入し，交通容量を空域容量一杯まで引き上げようという考え方がある。将来は，そのような方法が必要になるかもしれない。

本研究は，この精密航空管制の可能性を，AI（Artificial Intelligence,人工知能）の技法を用いて論じたものである。その一つのアプローチとして，コンフリクトフリー・プランニング・モデルなるものを試作した[2]~[5]。このモデルの作成には，GPSSによる交通流モデルを，先行モデルとして利用した。

2．現行管制方式

航空機が目的地まで飛ぶ方法に，VFR（有視界飛行方式）とIFR（計器飛行方式）がある。定期路線のフライトは，天候にかかわりなくほとんどがIFRで飛び，常時管制を受ける。

日本の管制機関が一日に扱うIFRフライト数は，約2,200である。内訳は，国内線が1,200，国内空港発着の国際線が400，残り600が軍用機と日本上空通過の民間機その他である。すなわち，わが国において，航空管制の対象となるフライトの大半は，時刻表に基づいて運航される定時性のトラヒックである。

ところが，その航空時刻表と鉄道における列車時刻表は，かなり性質が異なる。列車時刻表の背後には列車運行ダイヤグラムがあり，列車の運行は精密に制御されている。航空では，時刻表上のフライトはそのままでは管制の対象にならない。フライトの実行に先立って飛行計画の提出が必要である。そして出発準備ができた段階で，管制機関に管制承認を要求する。管制機関は，そのときの空域の状況と他の飛行計画とから，他機との管制間隔を考慮して，高度，出発時刻，経路の承認，すなわち管制承認を出す。フライトを開始した後は，担当管制官の間で次々と管制移管を受けながら，管制間隔を維持するための指示を受ける。

このような航空管制のやり方を列車制御の方法に近付けることはできないだろうか。すなわち，航空時刻表からあらかじめ全フライトの管制承認を作成し，これによる運航が可能であれば，管制の効率化と交通量増大の効果が期待できる。これについてAttwoollの主張がある。

3．Attwoollの考え

Attwoollは，航空交通の精密スケジュール化の問題について次のような考察をしている（1977年，英国）[6]。

航空管制の一つの理想的な形は，フライトの各々に対しその進行過程にタイム・スロットを割り当て，航空機をそのとおりに飛ばせることである。これにより，管制側遅延（ATC Delay）をゼロにし，交通量を理論的に可能な限界まで増加し得る。

しかし，このような理想的なシステムを考えても，フライトの発生がスケジュールからずれるとき，すなわち規則性からのランダム変動が生ずると，その効果は減少する。定量的には，ランダム変動の標準偏差がスロット間隔の値を越えると，その効果は無くなる。

当時の英仏間メインルートでは，整備上の障害，乗客の遅れ，気象の影響等に起因する会社側遅延（Company Delay）によるランダム変動の標準偏差が16分であり，滑走路のタイム・スロット間隔2分，航空路の4分に比べてはるかに大であり，現状ではこのようなシステムは実現し得ないとしている。

しかしながら，Attwoollはそこで，この魅力的な（attractive）システムを非現実なものとして否定すべきでない，その困難さを克服する努力を続けるならば，将来このようなシステムを実現することも可能であろうと述べている。

4．スケジューリング問題

このAttwoollの考えを実現するには，第一に航法の4次元高精度化，第二に運航管理の精密化，第三にダイヤの乱れへの対応の技術が必要である，そして，この三つの要件に先立って，列車ダイヤに相当するものを，航空管制においても作る必要がある。

列車ダイヤは（x,t）の2次元なので，紙の上に表現できるが，航空管制のダイヤは（x,y,z,t）の4次元になるので，これを列車ダイヤのように表現することはできない。しかしながら，この種の問題には，今日ではAIの技術のうち，エキスパートシステムによるスケジューリングの手法が適用できる。

そこで，航空時刻表を基に，あらかじめコンフリクトを除いた全フライトの管制承認を，エキスパートシステムを用いて作成することを試みた。このスケジューリングは，時刻表において潜在するコンフリクトを解消すると同時に，航空管制をシミュレーション・モデルで表現することになる。そこで，これをコンフリクトフリー・プランニング・モテル（Conflict-Free Planning Model）と呼ぶことにする。

5．コンフリクトフリー・プランニング・モテル

5.1　対象トラヒック

スケジューリングの対象を，GPSSモデルと同じ国内定期路線全便とした。入力データセットの構成もGPSSモデルのときと同じとし，FORTRANプログラムで前処理をして論理的エラーを除いておく。但し，数値コードの出力は今回は使用しない。

1989年8月の航空時刻表と，これに必要な航空路構成については，空港68，会社名6，機種12，フライト1,240，フィックス99，セグメント272，往復を別に数えてルート330であった。ルートを重ね合わせたものを図1に示す。

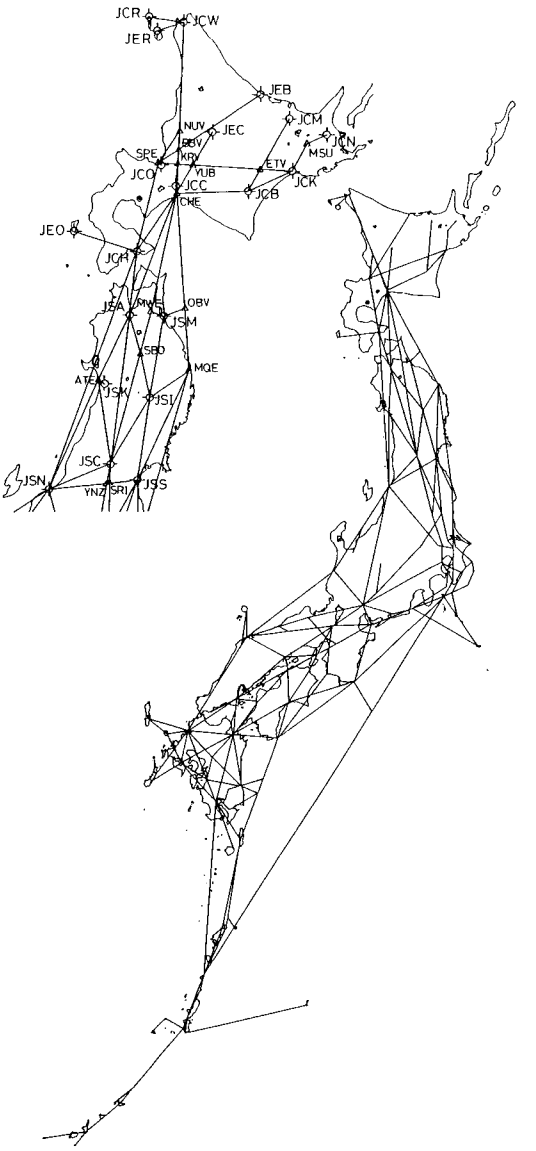


図1　航空路構成

5.2　エキスパートシステムESHELL

そのスケジューリングを行うために，LISPベースのエキスパートシステムESHELL[7],[8]を用いることにした。ESHELLは，プロダクションルール，フレーム，ブラックボード，及び推論エンジンで構成される。

プロダクションルールは，推論の基礎になる知識を，IF～THEN～の形式のルールとして表現する。いくつかのルールが知識源（KS: Knowledge Source）としてまとめられ，その知識源の集まりがプロダクションルールとなる。

フレームは，推論の前提となる静的な知識を階層構造で表現する。ブラックボード，すなわち黒

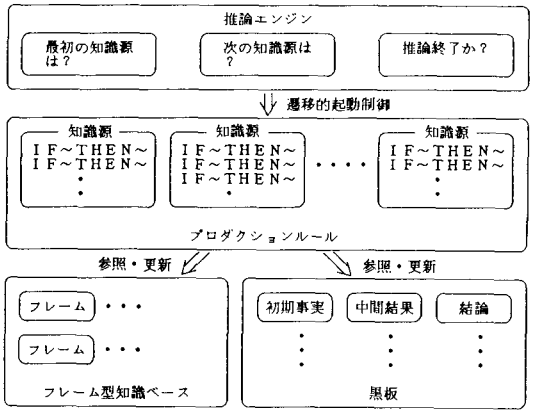


図2　ESHELLの推論方式

板は，推論のための初期事実，中間結果，結論を動的に書いたり，消したりする所である。推論エンジンは，知識源を遷移的に起動して，推論の進行をつかさどる。ESHELLの構成を図2に示す。

このESHELLで，どのようにしてスケジューリングを行うかは，プロダクションルールをどう構成するかという問題になる。今回のモデルではそれを図3のように設計した。

5.3　プロダクションルールの働き

先ず初期設定KSで時刻表と航空路データをフレームに格納する。この格納はESHELLとリンクするFORTRAN関数副プログラムを介して行う。その際，数値コードへの変換は不要で，記号と数値のままで入力される。格納はリスト形式*1でなされる。

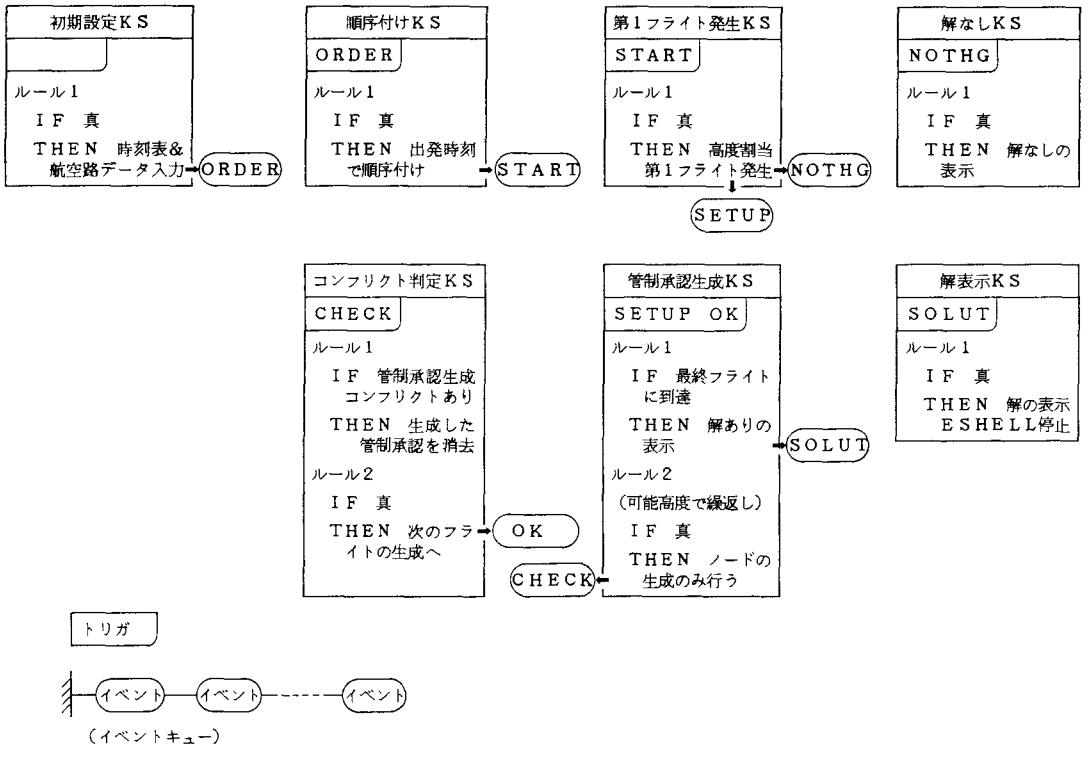


図3　プロダクションルールにおける知識源(KS)関連図

＊1　LISPではデータとプログラムを区別しない。どちらも，必要な要素を括弧で括った表現形式をとる。これをリストという。

次に順序付けKSで時刻表上の全フライトを出発時刻順に並び換える。そこで，第1フライト発生KSで，最初のフライトについて高度を与え，そのフライトの管制承認を，第1のノードとしてブラックボード上に作り出す。

第2のフライトについては，管制承認生成KSで管制承認を作成し，第1のフライトとの間の管制間隔を，コンフリクト判定KSでチェックする。コンフリクトがある場合は，高度を変更して管制承認を作り直す*2。

以下同様にして，先行フライトとのコンフリクトをチェックしながら，順次管制承認を作っていき，最終フライトに達することができれば，コンフリクトフリーな管制承認の集合が得られたことになる。

すなわち，高度選択を選択肢とし，コンフリクトの有無を制約条件として，バックトラッキング（backtracking, 後戻り戦略，枝刈り）により解を探索しようとするものである。図4にバックトラッキング法の概念を示す。

スケジューリング問題をFORTRAN等の手続型処理で解こうとすると，選択肢の全ての組合せをチェックすることになり，分岐は組合せ爆発（combinatorial explosion）を生じてしまい，プログラムを書くこと自体不可能である。

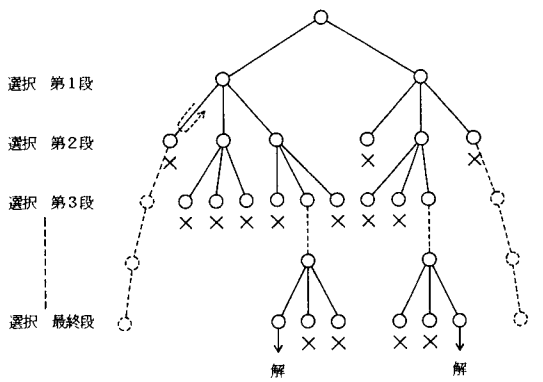


図4　バックトラッキング推論

LISP, Prolog等のAI言語，またそれらをベースにしたエキスパートシステムでは，このような問題について，バックトラッキング法で解の探索を行うことができる。なお，ESHELLではイベントドリブン方式という方法でバックトラックを起している。これについては付録1で述べる。

5.4　モデルの記述範囲

このモデルでのフライトの記述のうち，空港での動きは図5に示すものとした。すなわち，出発においては，牽引，地上滑走，離陸待ち，滑走路占有ののち離陸とした。到着においては，着陸待ちののち着陸，滑走路占有，地上滑走をして駐機場に入るものとした。

エンルート飛行は，図6に示すように，上昇，巡航，下降から成るものとした。巡航高度は，飛行方向，距離，機種の組合せで決まる2～3のフライト・レベルの中で，高い方から選択するものとした。高度変化率は一律に，ターボプロップ機で1,000フィート／分，ジェット機で2,000フィート／分とした。

コンフリクト・チェックは，図7に示すように，巡航部分における同一高度での他機との一定距離以内の接近の有無と，フィックスでの同一高度同一時刻通過の有無に限定した。

5.5　プログラミング

ESHELLのプロダクションルールを上記のように構成したところ，知識源7，その中のIF～THEN～ルールは9，細部を記述するLISPによるユーザー定義関数は127となった。

ユーザー定義関数の中から，コンフリクト・チェックにかかわる2つの関数を，図8に示す。LISPが関数型言語であることに注意しておく。

関数US:CONFLICT-CHECKは，「新しく生成された管制承認FRESH-CLEARANCEについてコンフリクト・チェックをせよ。それは，相手方WORKを探し出し，両者について関数US:CHECK-CLR-TO-CLRを用いてチェックを行うことである」と宣言して，この関数はここで文法的に閉じる。

同様に，関数US:CHECK-CLR-TO-CLRは，巡航部分でのチェックUS:CHECK-SEG-TO-SEGとフィックスでのチェックUS:CHECK-FIX-TO-

---

＊2　実際には，プログラミング上，管制承認KSでは高度選択の選択肢だけを出しておく。コンフリクト・チェックKSで管制承認の作成とコンフリクト・チェックを行う。

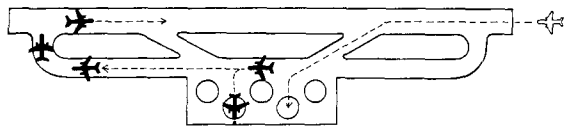


図5　滑走路オペレーション・モデル

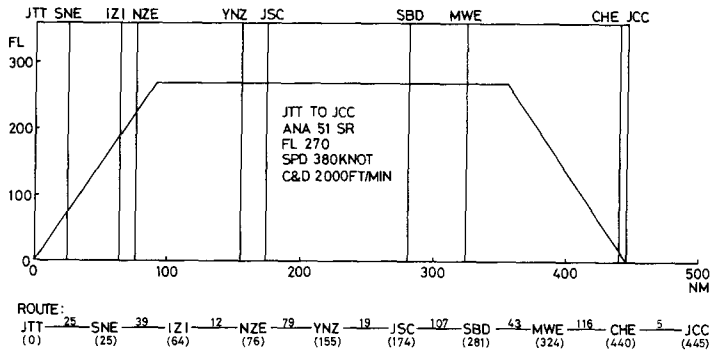


図6　エンルート飛行モデル

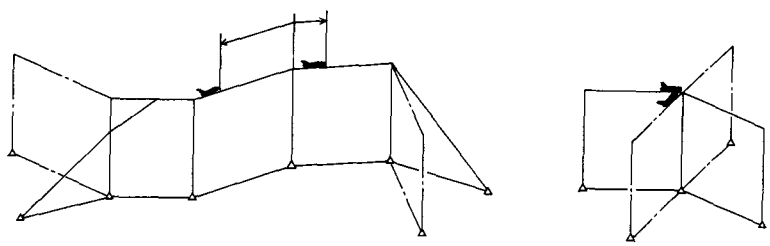


図7　コンフリクト・チェック・モデル

```
(DEFUN US:CONFLICT-CHECK  (FRESH-CLEARANCE)
    (DO((WORK (@VALUE FRESH-CLEARANCE 'LAST) (@VALUE WORK 'LAST)))
        ((NULL WORK) NIL)
        (COND((US:CHECK-CLR-TO-CLR FRESH-CLEARANCE WORK) (EXIT T)))))

(DEFUN US:CHECK-CLR-TO-CLR  (FCLR LCLR)
    (LETS  ((FALT (@VALUE FCLR 'ALTITUDE))
            (LALT (@VALUE LCLR 'ALTITUDE)))
     (COND((OR (EQ FALT 0) (EQ LALT 0) (NEQ FALT LALT)) NIL)
           (T(LETS  ( (FSEG
                        (US:REMOVE-NIL (US:SEGMENT-SET FCLR FALT)))

                      (LSEG
                        (US:REMOVE-NIL (US:SEGMENT-SET LCLR LALT)))
                     (FFIX (US:REMOVE-NIL (US:FIX-SET FCLR FALT)))
                     (LFIX (US:REMOVE-NIL (US:FIX-SET LCLR LALT))))
              (COND((AND (NULL (US:SEG-INTERSECTION FSEG LSEG))
                         (NULL (US:FIX-INTERSECTION FFIX LFIX)))
                    NIL)
                   ((US:CHECK-SEG-TO-SEG FSEG FALT LSEG LALT) T)
                   ((US:CHECK-FIX-TO-FIX FFIX FALT LFIX LALT) T)
                   (T NIL)))))))
```

図8　ユーザー定義関数の一部

FIXを行うことを宣言して，これもここで閉じる。

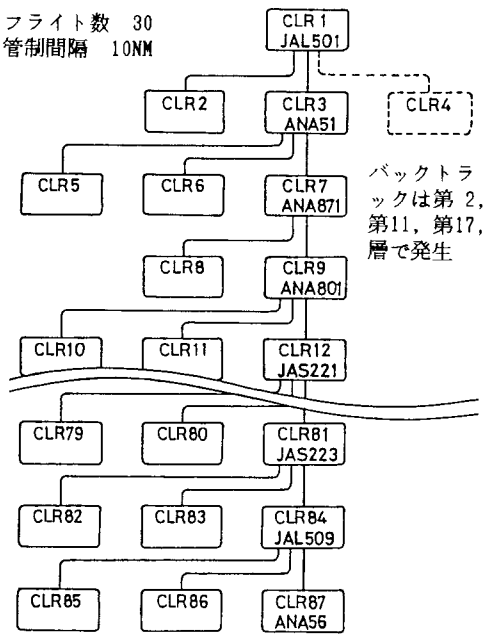
このような関数型言語を用いて作ったエキスパートシステムでは，モデルの改変や機能追加は，関数の差換や追加で済む。手続型言語では，全体設計を先に固めてから細部を記述しているので，一度出来上がったものを修正するのは容易でない。

上記プロダクションルールに，ブラックボードの設計と推論エンジンの作動指定パラメータを加えたものを，ESHELLではプロダクション型知識ベースと呼んでいる。そのリスト全体を付録2に示す。また，本モデルでのフレーム構成を付録3に示す。

6．テスト・ランの結果

6.1 30フライトでのテスト・ラン

上記モデルにおいて，先ずトラヒックを30フライトにしてテスト・ランを行った。この30フライトは羽田と日本北東部主要空港の間の午前中のトラヒックを基にした。



ノードの名称を示すCLRはATC－CLEARANCE（管制承認）を意味し，続く番号はそのノードの生成順序を示す。

図9　テスト・ランにおける探索結果(部分)

管制間隔は10NMとした。トラヒックが少ないのと，コンフリクト・チェックの条件が緩いので，通常のレーダー管制間隔3～5 NMより長くした。図9に示すように，3回バックトラックを起し，コンフリクトフリーな30の管制承認の集合，すなわち解を得た。

第2層で残ったノードを図10に示す。これは，羽田から千歳に向かうANA51便で，当初29,000フィートで管制承認を作成したが，先行フライトとコンフリクトがあり，デリートしてバックトラック，高度27,000で管制承認が作成された。フライトの進行過程は，図5及び図6との照合により理解されよう。

高度25,000のノードは，下の方の層ですべての選択肢が切られた場合に，ここまで戻ってきてバックトラックを起すためのものである。

問題は，この解を得るに要した時間である。4 MIPSのメインフレーマFACOM-M360APを，6 M BITE領域のTSSで使用して，CUPタイム2時間50分を必要とした。但し，LISPはインタープリタでの使用である。

6.2 1,240フライトのラン・タイム見積

フルサイズ1,240フライトのトラヒックを入力した場合は，3時間で第6層まで到達した。この間バックトラックが1回発生している。

このフルサイズ・モデルがどの位の時間でランできるかを概略見積った。LISPのコンパイル，高速のワークステーションと効率の良いエキスパートシステムの使用，プログラムの改良等で，10分のオーダーでランできるとの見通しを得た。その見積りは付録4で述べる。

6.3 解の存在の可能性

上記テスト・ランにおいて，30フライトのトラヒックについては解を得ることができた。しかし，1,240フライトについては，この方法で解が得られるかどうかは，今のところ分らない。選択肢が高々3では，1,240層もの探索で解を得ることは難しいかも知れない。何らかのヒューリスティック（heuristic,発見的）な方法が必要かもしれない。例えば，幹線のフライトについて解を得たのち，ローカルを挿入するというような方法もある。

プログラム上は，高度の選択でコンフリクトが

```
NODE NAME <ATC-CLEARANCE2>
  PROTO: ATC-CLEARANCE
  LEVEL      = (250)
  DEFERMENT = (0)
  LAST = (ATC-CLEARANCE1)
NODE NAME <ATC-CLEARANCE3>
  PROTO: ATC-CLEARANCE
  ORDER                          = (2)
  LEVEL                          = (270)
  DEFERMENT                      = (0)
  ROUTE                          = ((JTT TO JCC LONG 445))
  AIRLINE                        = (ANA)
  FLIGHT-NUMBER                  = (51)
  TYPE                           = ((SR TYPE3))
  DEPARTURE-AIRPORT              = (JTT)
  ARRIVAL-AIRPORT                = (JCC)
  DEPARTURE-TIME                 = (700)
  ARRIVAL-TIME                   = (825)
  ALTITUDE                       = (270)
  SPEED                          = (380)
  TOWING                         = ((700 704))
  TAXIING-BEFORE-TAKE-OFF        = ((704 710))
  RUNWAY-OCCUPANCY-FOR-TAKE-OFF  = ((710 711))
  FLIGHT-ON-EACH-SEGMENT         = ((((JTT SNE) (711 0) (715 80))
                                    ((SNE IZI) (715 80) (721 190))
                                    ((IZI NZE) (721 190) (723 230))
                                    ((NZE YNZ) (723 230) (725 270)
                                     (735 270))
                                    ((YNZ JSC) (735 270) (738 270))
                                    ((JSC SBD) (738 270) (755 270))
                                    ((SBD MWE) (755 270) (802 270))
                                    ((MWE CHE) (802 270) (807 270)
                                     (820 20))
                                    ((CHE JCC) (820 20) (821 0))))
  RUNWAY-OCCUPANCY-FOR-LANDING   = ((821 823))
  TAXIING-AFTER-LANDING          = ((823 825))
  LAST = (ATC-CLEARANCE1)
```

図10　第2層で残ったノード

解消しないとき，出発時刻の方も一定時間刻みで一定限度まで延伸できるようにしてある。すなわち，高度と出発延伸の組合せで選択肢を増やすこともできる。今回のテスト・ランではその機能は使用していないが，これより探索の範囲を広げることも可能である。

一方，解が複数存在することもあり得る。そのときは，全フライトの離着陸待ち時間の総和を最小にするものといった評価基準を設けて，最適の解を取り出す探索法も考えられる。

6.4　同時並行処理記述の可能性

このコンフリクトフリー・プランニング・モデルは，GPSSによる航空交通流モデルを基にした。しかし，GPSSで表現した着陸優先による滑走路使用は，今回のモデルでは取り入れていない。これを表現するには，GPSSの基本機能であるオバラルスキャンを記述する必要がある。ESHELLにも，イクスペクテーションといって，事象の発生を一時保留し，後続事象の発生によってその保留を解除するという機能もあるが，今回はそこまで記述しなかった。今回のモデルでは，各滑走路の占有はそこでの離着陸を含めてのフライトの生成順としている。

GPSSは同時並行現象の記述には適しているが，スケジューリングの機能はない。エキスパートシステムはスケジューリングには適しているが，同時並行現象を記述しようとすると面倒になる。航空管制のような，大規模な同時並行処理システムでスケジューリング問題を解こうとするとき，ツールの選択，またその組合せは研究課題になる。

7．おわりに

この研究の今後の課題は，何よりも実行時間を

短縮し，フルサイズ・モデルでの解の探索を試みることである。また，モデルの記述を詳細にすること，すなわち，管制方式基準と管制官の経験的知識をどう取り入れるかについても検討を要する。これはエキスパートシステム構築での知識獲得に当る。

Attwoollによる精密航空管制を実現するには，このスケジューリングに加え，前述の三つの技術が必要である。第一の航法の高精度化については，ICAO（国際民間航空機関）のFANS（Future Air Navigation Systems）構想によりその可能性がでてきた。第二の運航管理の精密化については，航空会社で現用しているエキスパートシステムにその兆しを見ることができる。第三のダイヤの乱れへの対応については，リアルタイム性に主眼を置いた研究開発も行われている。これらの技術と，航空管制へのAI適用の動向，必要性，問題点等については，文献(9)～(11)を参照されたい。

このような技術は，将来，航空管制の自動化に道を開くことになるであろう。しかしそれは，管制官の存在を否定するものではない。そこでは，人間と機械の調和が最も重要な課題になるであろう。

参考文献

(1) 仁科光雄：“離散型シミュレーション言語GPSSによる航空交通流モデル”，電子航法研究所報告，No.77，pp.1-25，1993年9月

(2) 仁科光雄：“知識情報処理による管制承認の作成について”，第21回電子航法研究所研究発表会講演概要，pp.17-20，1989年5月

(3) 仁科光雄：“知識情報処理による管制承認の作成（その2）”，第24回電子航法研究所研究発表会講演概要，pp.61-64，1992年5月

(4) 仁科光雄：“エキスパートシステムによる精密航空管制の可能性について”1992年電子情報通信学会春季大会，B-161，p.2-161，1992年3月

(5) 仁科光雄：“Attwoollによる精密航空管制の考え方とコンフリクトフリー・プランニングについて”，電子情報通信学会技術研究報告，SANE92-36，pp.9-15，1992年9月

(6) Attwooll，V. W.：“Some Mathematical Aspects of Air Traffic Systems”，The Journal of Navigation，Vol.30，No.3，pp394-414（Sep. 1977）

(7) “FACOM ESHELL解説書（第2版）”富士通株式会社，1986年9月

(8) “OS IV ESHELLの適用”，富士通株式会社情報システムボラトリ，1986年6月

(9) 片野忠夫：“航空航法システムの将来展望”，電子情報通信学会技術研究報告，SANE92-19，pp.15-22，1992年7月

(10) 仁科光雄：“航空交通流モデルについて－航空管制，シミュレーション，そしてAI－”，オペレーションズ・リサーチ，Vol.35，No.2，pp.107-113，1990年2月

(11) 仁科光雄：“航空管制と知識情報処理について”，航海，113号，pp.21-28，1992年9月

付録1　イベントドリブン方式によるバックトラッキング法

ESHLLでのイベントドリブン方式によるバックトラッキング法を説明する。本モデルでは単純な縦型探索を行う。次の二つの原則により探索を行うように，推論エンジンの作動指定パラメータを設定する。

第一に，知識源（KS）内のルールの実行時にイベントが発行された場合，そのイベントは，イベントキューの最後尾につながれる。第二に，一つのKSの動作が終了すると，推論エンジンはイベントキューから最後尾のイベントを取り出し，そのイベント名と同じトリガを持つKSを次に起動する。

この原則を本文図3に適用すれば，探索の過程において，付図1のようにイベントがキューの中に残っていくことが分る。

NOTHGイベントは最後まで残る。これが取り出されるのは，解が存在しない場合である。CHECKイベントが2つ連続して取り出されて，バックトラックが1回発生する。

管制承認生成KSでCHECKイベントが発行されるとき，そこで生成されたノードの名称（ATC-CLEARANCE n）がそのイベントにフォーカスノードとして記録される。

コンフリクト判定KSでCHECKイベントが取り出されたとき，フォーカスノードは@FOCUS-NODE（後出の付図2プログラム・リスト中，＊＊KNOWLEDGE SOURCEの設計＊＊に注意）

として参照できる。これにより，どのノードについてコンフリクト・チェックが行われるかが分る。

通常は，イベントキューが空になると，推論動作は終了する。このモデルでは，NOTHGかSOLUT，どちらかのイベントが取り出されると，動作が終了するようにしてある。

本文中図4と図9が裏返しになっているのは，選択肢すなわちノードの作成が高度の低い方からなされるにもかかわらず，コンフリクト・チェックが高度の高い方から行われるためである。バックトラッキングの本質には関係ない。

付録2　プロダクション型知識ベースのプログラム・リスト

付図2にブラックボード，プロダクションルール，推論エンジン（まとめてプロダクション型知識ベース）のプログラム・リストを示す。

付録3　フレーム構成

付図3にフレーム構成を示す。本モデルではフレームを3層に構成した。例えば，AIR-TRAFFIC-DATA～AIRPORT-LIST～AIRPORTというリンクを見ることができる（プログラム・リストの出力はアルファベット順であるが，LINKをたどれば分る）。

付図4は，初期設定KSが作動して入力データが読み込まれたのちのフレームの一部を示す。付図3の最下位概念のインスタンスフレームを鋳型として，その複製を作りながら，データを格納していることに注意。

付録4　フルサイズ・モデルのラン・タイム見積りについて

LISPは，プログラムの作成とデバッグをインタープリタで行う。手続型言語に比べて，プログラム開発の効率は良いが，LISP固有の性質から実行速度は遅い。出来上がったプログラムをコンパイルしてから実行すれば，かなり速くなる。プログラムにもよるが，速度が10倍位になることもある。

上記のテスト・ランでは，4 MIPSのメインフレーマを使用した。今日ではこのような問題にはワークステーションを使用する方が良い。最高速のものでは100MIPSといったものもあり，上記メインフレーマの10倍程度速いものは容易に得られる。

ソフトの面でも改良がなされている。今回使用したESHELLは，わが国でAIの実用化が始まった1980年代半ばにレリーズされたものである。今となっては古いツールである。現在ではESHELLの10倍程速いツールもある。

すなわち，コンパイルの効果、高速のハード，適切なソフトの選択で，実行速度は上記テスト・ランの1,000倍程度は得られよう。

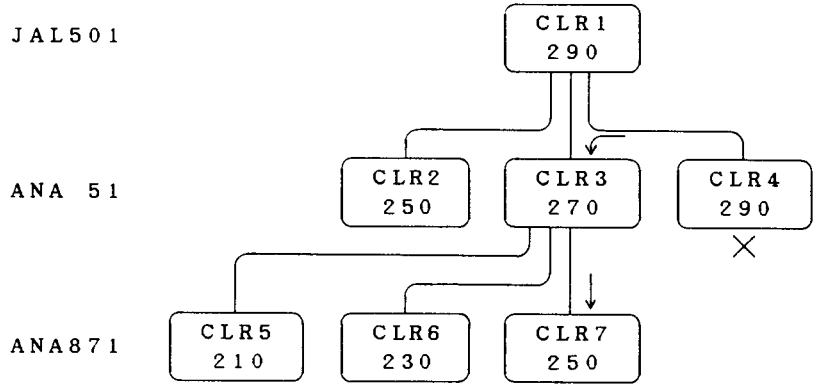
さて，1,240フライトのランでは，第6層に達するのに3時間を要している。解が存在するとして最深層に達するのに，単純計算で200倍として，600時間が必要となる。そこで，上記の見積りで1,000倍の速度が得られるとすれば，フルサイズのランに要する時間は0.6時間となる。

但し，単純計算の仮定に問題がある。現在のプログラムでは，生成した管制承認の各々について，その先行フライトのすべてとコンフリクトを調べている。コンフリクト・チェックの回数は深い層にいくほど多くなり，ラン全体では1,240から2つをとる組合せの総数77万回にもなる。単純に所要時間が200倍というわけにはいかない。

しかし，これはヒューリスティックな方法で回避できる。国内線フライトの飛行時間は高々3時間である。一つの管制承認からみて，すでにフライトを終了しているものとの間ではコンフリクト・チェックの必要はない。また，わが国のトラヒックの性質から，地域の異なるローカル線相互間ではコンフリクト・チェックを省いてよい。このような方法で，コンフリクト・チェックの回数はかなり押えることができるであろう。

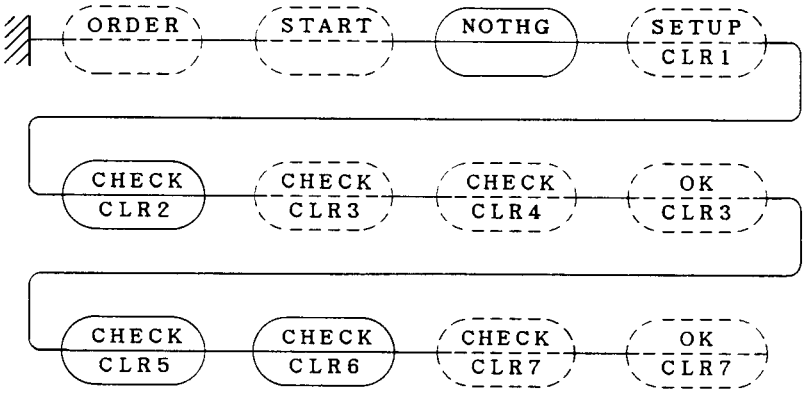
また，現行のプログラムでは，フレームの使用が適切でない。フレームは元来，知識の間でのインヘリタンス（inheritance, 属性の継承）を可能にするためのものである。しかし，本モデルではこの機能はほとんど使用していない。フレーム構成はもっと簡単でよい。これにより，この種のプログラムでラン・タイムの大半を占める索表時間を大幅に削減できる。

ヒューリスティックな方法とプログラムの改良で，0.6時間の見積りをさらに縮めることができれば，1,240フライトのフルサイズ・モデルを10分のオーダーでランすることも可能であろう。



第3層までの探索

（ＣＬＲｎの下の数値はフライト・レベルを示す）



第3層までの探索におけるイベントキューの消息

（実線はキューに残ったイベント，破線は一度つながれたのち取り出されたイベントを示す。イベント内の上段はイベント名，下段はフォーカスノード名を示す）

付図1　イベントドリブン方式縦型探索

```
*** ESHELL USER PROGRAM 91/09/06 14:03:59 ***

** BLACKBOARDの設計 **

LEVEL名 : @DATALEVEL
ATTRIBUTES: END-FLIGHT SEPARATION-STANDARD DEPARTURE-DEFERMENT FLIGHT-LEVELS GROUND~SERVICE-TIME-MEANS TOWING-TIME-MATRIX
            TAKE-OFF-TAXI~TIME-MATRIX TAKE-OFF-RUNWAY-TIME-MATRIX LANDING-RUNWAY-TIME-MATRIX LANDING-TAXI-TIME-MATRIX
STATIC NODE <@DATA>
DEFAULT EXPRESSION:
  END-FLIGHT                    =  (LENGTH
                                      (@FGET 'FLIGHT-SCHEDULE-LIST
                                        'INSTANCE '¥LINK))
  SEPARATION-STANDARD           =  (RIND
                                     "INPUT SEPARATION STANDARD IN NM =>")
  DEPARTURE-DEFERMENT           =  (RIND
                                     "INPUT APPROVAL DEPARTURE-DEFERMENT IN MIN IN LIST =>")
  FLIGHT-LEVELS                 =  '( (((50 70) (40 60))
                                       ((70 90) (60 80))
                                       ((90 110) (80 100)))

                                      (((190 210) (200 220))
                                        ((210 230 250) (220 240 260))
                                        ((250 270 290) (260 280 310)))

                                      (((190 210) (200 220))
                                        ((210 230 250) (220 240 260))
                                        ((250 270 290) (260 280 310))))
  GROUND-SERVICE-TIME-MEANS     =  '((0.0 3.0 3.0) (3.0 3.0 5.0)
                                     (1.5 1.5 1.5) (1.5 1.5 1.5)
                                     (3.0 3.0 3.0))
  TOWING-TIME-MATRIX            =  '(((100 0)) ((20 2) (80 3) (100 4))
                                      ((20 2) (80 3) (100 4)))
  TAKE-OFF-TAXI-TIME-MATRIX     =  '(((30 2) (70 3) (100 4))
                                     ((30 2) (70 3) (100 4))
                                     ((30 4) (70 5) (100 6)))
  TAKE-OFF-RUNWAY-TIME-MATRIX =  '(((50 1) (100 2)) ((50 1) (100 2))
                                     ((50 1) (100 2)))
  LANDING-RUNWAY-TIME-MATRIX    =  '(((50 1) (100 2)) ((50 1) (100 2))
                                     ((50 1) (100 2)))
  LANDING-TAXI-TIME-MATRIX      =  '(((33 2) (67 3) (100 4))
                                     ((33 2) (67 3) (100 4))
                                     ((33 2) (67 3) (100 4)))

LEVEL名 : ATC-CLEARANCE
ATTRIBUTES: ORDER LEVEL DEFERMENT ROUTE AIRLINE FLIGHT-NUMBER TYPE DEPARTURE-AIRPORT ARRIVAL-AIRPORT DEPARTURE-TIME
            ARRIVAL-TIME ALTITUDE SPEED TOWING TAXIING-BEFORE-TAKE-OFF RUNWAY-OCCUPANCY-FOR-TAKE-OFF FLIGHT-ON-EACH-SEGMENT
            RUNWAY-OCCUPANCY-FOR-LANDING TAXIING-AFTER-LANDING
LINKS     : LAST
DEFAULT EXPRESSION:
  ORDER                         =  (ADD1 (@VALUES @NODE 'LAST 'ORDER))
  ROUTE                         =  (US:ROUTE? (@VALUE @NODE 'ORDER))
  AIRLINE                       =  (US:AIRLINE? (@VALUE @NODE 'ORDER))
  FLIGHT-NUMBER                 =  (US:FLIGHT~NUMBER?
                                     (@VALUE @NODE 'ORDER))
  TYPE                          =  (US:TYPE? (@VALUE @NODE 'ORDER))
  DEPARTURE-AIRPORT             =  (US:DEPARTURE-AIRPORT?
                                     (@VALUE @NODE 'ORDER))
  ARRIVAL-AIRPORT               =  (US:ARRIVAL-AIRPORT?
                                     (@VALUE @NODE 'ORDER))
  DEPARTURE-TIME                =  (US:DEPARTURE-TIME?
                                     (@VALUE @NODE 'ORDER))
  ARRIVAL-TIME                  =  (US:ARRIVAL-TIME?
                                     (@VALUE @NODE 'ORDER))
  ALTITUDE                      =  (US:ALTITUDE-IN-DEFAULT?
                                     (@VALUE @NODE 'LEVEL))
  SPEED                         =  (US:SPEED?)
  TOWING                        =  (US:TOWING?
                                     (@VALUE @NODE 'DEFERMENT))
  TAXIING-BEFORE-TAKE-OFF       =  (US:TAXIING-BEFORE-TAKE-OFF?)
  RUNWAY-OCCUPANCY-FOR-TAKE-OFF =  (US:RUNWAY-OCCUPANCY-FOR-TAKE-OFF?
                                     (@VALUE @NODE 'LAST))
  FLIGHT-ON-EACH-SEGMENT        =  (US:FLIGHT-ON-EACH-SEGMENT?)
  RUNWAY-OCCUPANCY-FOR-LANDING  =  (US:RUNWAY-OCCUPANCY-FOR-LANDING?
                                     (@VALUE @NODE 'LAST))
  TAXIING-AFTER-LANDING         =  (US:TAXIING-AFTER-LANDING?)

** KNOWLEDGE SOURCE の設計 **

KS NAME : INITIALIZE-KS (FORWARD)
STRATEGY: SINGLE ONCEONLY
RULE# <1>
 IF
   '(READ-IN AIR~TRAFFIC-DATA)
 THEN
  1 EXECUTE
    EXPRESSION :  (US:READ-AIRPORT-LIST)
  2 EXECUTE
    EXPRESSION :  (US:READ-AIRLINE-LIST)
  3 EXECUTE
    EXPRESSION :  (US:READ-TYPE-LIST)
  4 EXECUTE
    EXPRESSION :  (US:READ-FLIGHT-SCHEDULE-LIST)
  5 EXECUTE
    EXPRESSION :  (US:READ-IDLE-LINE-LIST)
  6 EXECUTE
    EXPRESSION :  (US:READ-FIX-LIST)
  7 EXECUTE
    EXPRESSION :  (US:READ-SEGMENT-LIST)
  8 EXECUTE
    EXPRESSION :  (US:READ-ROUTE-LIST)
  9 EXECUTE
```

付図２－１　プロダクション型知識ベース(1)

```
    EXPRESSION :  (US:INITIALIZE-RANDOM-NUMBER-GENERATOR)
  10 PROPOSE
    CHANGE TYPE:  NULL
    EVENT NAME :  ORDER
    NODE       :  NIL

KS NAME : ORDERING-KS (FORWARD)
TRIGGERS: ORDER
STRATEGY: SINGLE ONCEONLY
RULE# <1>
 IF
   '(ORDERING FLIGHT-SCHEDULE)
 THEN
  1 EXECUTE
    EXPRESSION :  (US:ORDERING-FLIGHT-SCHEDULE)
  2 PROPOSE
    CHANGE TYPE:  NULL
    EVENT NAME :  START
    NODE       :  NIL

KS NAME : FIRST-FLIGHT-KS (FORWARD)
TRIGGERS: START
STRATEGY: SINGLE ONCEONLY
LOCALS  : ENDFLT =  (@VALUE '@DATA 'END-FLIGHT)
          SEPSTD =  (@VALUE '@DATA 'SEPARATION-STANDARD)
          DEPDEF =  (@VALUE '@DATA 'DEPARTURE-DEFERMENT)
RULE# <1>
 IF
   '(GENERATE ATC-CLEARANCE OF FIRST FLIGHT)
 THEN
  1 EXECUTE
    EXPRESSION :  (FORMAT "BEGIN SEARCH OF SOLUTION OF /C FLIGHTS /N"
                    ENDFLT)
  2 EXECUTE
    EXPRESSION :  (FORMAT "IN SEPARATION-STANDARD /C NM /N" SEPSTD)
  3 EXECUTE
    EXPRESSION :  (FORMAT "IN APPROVAL DEPARTURE-DEFERMENT /C MIN /N"
                    DEPDEF)
  4 PROPOSE
    CHANGE TYPE:  NULL
    EVENT NAME :  NOTHING
    NODE       :  NIL
  5 PROPOSE
    CHANGE TYPE:  ADD
    EVENT NAME :  SETUP
    LEVEL      :  ATC-CLEARANCE
    ATTRIBUTES : ORDER                          =  1
                 ROUTE                          =  (US:ROUTE? 1)
                 AIRLINE                        =  (US:AIRLINE? 1)
                 FLIGHT-NUMBER                  =  (US:FLIGHT-NUMBER?
                                                     1)
                 TYPE                           =  (US:TYPE? 1)
                 DEPARTURE-AIRPORT              =  ( US:DEPARTURE-AIRPORT?
                                                     1)
                 ARRIVAL-AIRPORT                =  (US:ARRIVAL-AIRPORT?
                                                     1)
                 DEPARTURE-TIME                 =  (US:DEPARTURE-TIME?
                                                     1)
                 ARRIVAL-TIME                   =  (US:ARRIVAL-TIME? 1)
                 SPEED                          =  (US:SPEED?)
                 ALTITUDE                       =  (US:ALTITUDE?)
                 TOWING                         =  (US:TOWING? 0)
                 TAXIING-BEFORE-TAKE-OFF        =  (US:TAXIING-BEFORE-TAKE-OFF?)
                 RUNWAY-OCCUPANCY-FOR-TAKE-OFF  =  ( US:RUNWAY-OCCUPANCY-FOR-TAKE-OFF?
                                                     NIL)
                 FLIGHT-ON-EACH-SEGMENT         =  (US:FLIGHT-ON-EACH-SEGMENT?)
                 RUNWAY-OCCUPANCY-FOR-LANDING   =  ( US:RUNWAY-OCCUPANCY-FOR-LANDING?
                                                     NIL)
                 TAXIING-AFTER-LANDING          =  (US:TAXIING-AFTER-LANDING?)
    LINKS      : LAST =  NIL

KS NAME : SETUP-CLEARANCE-KS (FORWARD)
TRIGGERS: SETUP OK
STRATEGY: SINGLE ONCEONLY
LOCALS  : LAST-CLEARANCE =  @FOCUSNODE
RULE# <1>
 IF
   (EQ (@VALUE LAST-CLEARANCE 'ORDER) (@VALUE '@DATA 'END-FLIGHT))
 THEN
  1 EXECUTE
    EXPRESSION :  (FORMAT "THE AI HAS FOUND A SOLUTION /N")
  2 PROPOSE
    CHANGE TYPE:  NULL
    EVENT NAME :  SOLUTION
    NODE       :  LAST-CLEARANCE
RULE# <2>
 ITERATION: FL =  (US:FLIGHT-LEVELS
                     (US:ORDER-PLUS-ONE LAST-CLEARANCE))
            DD =  (@VALUE '@DATA 'DEPARTURE-DEFERMENT)
 IF
   '(GENERATE BRANCH)
 THEN
  1 PROPOSE
    CHANGE TYPE:  ADD
    EVENT NAME :  CHECK
    LEVEL      :  ATC-CLEARANCE
    ATTRIBUTES : LEVEL     =  FL
                 DEFERMENT =  DD
    LINKS      : LAST =  LAST-CLEARANCE

KS NAME : CONFLICT-CHECK-KS (FORWARD)
```

付図２－２　プロダクション型知識ベース(2)

```
TRIGGERS: CHECK
STRATEGY: SINGLE ONCEONLY
LOCALS  : FRESH-CLEARANCE =  @FOCUSNODE
RULE# <1>
 IF
   (PROGN (@VALUE FRESH-CLEARANCE 'ORDER)
          (@VALUE FRESH-CLEARANCE 'ROUTE)
          (@VALUE FRESH-CLEARANCE 'AIRLINE)
          (@VALUE FRESH-CLEARANCE 'FLIGHT-NUMBER)
          (@VALUE FRESH-CLEARANCE 'TYPE)
          (@VALUE FRESH-CLEARANCE 'DEPARTURE-AIRPORT)
          (@VALUE FRESH-CLEARANCE 'ARRIVAL-AIRPORT)
          (@VALUE FRESH-CLEARANCE 'DEPARTURE-TIME)
          (@VALUE FRESH-CLEARANCE 'ARRIVAL-TIME)
          (@VALUE FRESH-CLEARANCE 'SPEED)
          (@VALUE FRESH-CLEARANCE 'ALTITUDE)
          (@VALUE FRESH-CLEARANCE 'TOWING)
          (@VALUE FRESH-CLEARANCE 'TAXIING-BEFORE-TAKE-OFF)
          (@VALUE FRESH-CLEARANCE 'RUNWAY-OCCUPANCY-FOR-TAKE-OFF)
          (@VALUE FRESH-CLEARANCE 'FLIGHT-ON-EACH-SEGMENT)
          (@VALUE FRESH-CLEARANCE 'RUNWAY-OCCUPANCY-FOR-LANDING)
          (@VALUE FRESH-CLEARANCE 'TAXIING-AFTER-LANDING)
          (US:CONFLICT-CHECK FRESH-CLEARANCE))
 THEN
  1 EXECUTE
    EXPRESSION :  (@DELNODE FRESH-CLEARANCE)
RULE# <2>
 IF
   '(FRESH-CLEARANCE OK)
 THEN
  1 PROPOSE
    CHANGE TYPE:  NULL
    EVENT NAME :  OK
    NODE       :  FRESH-CLEARANCE

KS NAME : NO-SOLUTION-KS (FORWARD)
TRIGGERS: NOTHING
STRATEGY: SINGLE ONCEONLY
RULE# <1>
 IF
   '(NO SOLUTION)
 THEN
  1 EXECUTE
    EXPRESSION :  (FORMAT "THERE IS NO SOLUTION /N")

KS NAME : SOLUTION-OUTPUT-KS (FORWARD)
TRIGGERS: SOLUTION
STRATEGY: SINGLE ONCEONLY
LOCALS  : END-CLEARANCE =  @FOCUSNODE
RULE# <1>
 IF
   '(OUTPUT SOLUTION)
 THEN
  1 EXECUTE
    EXPRESSION :  (US:SOLUTION-OUTPUT END-CLEARANCE)
  2 PROPOSE
    CHANGE TYPE:  NULL
    EVENT NAME :  TERM
    NODE       :  NIL

** CONTROL STRUCTURE の設計 **

INITIALIZATION FUNCTION =  @INITIALIZE
TERMINATION FUNCTION    =  (LAMBDA NIL
                              (OR (NULL @EVENTQUEUE)
                                (ASSQ 'TERM @EVENTQUEUE)))
STEP選択ルール
  EXPECTATION <=  @EXPECTATIONQUEUE
  EVENT       <=  @EVENTQUEUE
EVENT制御情報
  SELECTION METHOD =  @DEPTHFIRST
  COLLECTION RULES =
EXPECTATION制御情報
  SELECTION METHOD =  @BREADTHFIRST
  MATCHER          =  @AND
LHS EVALUATOR      =  @AND
VALUE ADJUSTER     =  @ASIS
POSTPROCESSING FUNCTION = US:POSTPROCESS

** ユーザ定義関数 **

 (DEFUN US:READ-AIRPORT-LIST  NIL
     (CATCH  'ENDMARK
        (LOOP(LOOP (SETQ ATFBF (SUBATF "DDD"))
                   (SETQ  APONR
                          (US:STRING-TO-NUMBER
                           (SUBSTRING ATFBF 0 3))
                          APORT
                          (INTERN (SUBSTRING ATFBF 4 7))
                          APONM
                          (INTERN
                            (US:REMOVE-EXTRA-SPACE
                              (SUBSTRING ATFBF 8 24))))
                   (COND((EQUAL APONR 999) (THROW 'ENDMARK NIL))
                         ((NOT (EQUAL APONR 888)) (EXIT NIL))
                         (T NIL)))
              (LETS  ((FRAME (@FINSTANTIATE 'AIRPORT-LIST 'AIRPORT)))
               (@FPUT FRAME 'NUMBER '¥VALUE APONR)
               (@FPUT FRAME 'ABBREV '¥VALUE APORT)
               (@FPUT FRAME 'NAME '¥VALUE APONM))))
     (@FDESTROY 'AIRPORT)
```

付図２－３　プロダクション型知識ベース(3)

```
    (US:POSTPROCESS-READ-LIST 'AIRPORT-LIST))

(DEFUN US:READ-AIRLINE-LIST  NIL
    (CATCH  'ENDMARK
       (LOOP(LOOP (SETQ ATFBF (SUBATF "DDD"))
                  (SETQ  ALNNR
                         (US:STRING-TO-NUMBER
                          (SUBSTRING ATFBF 0 3))
                         ALN
                         (INTERN (SUBSTRING ATFBF 4 7))
                         ALNNM
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 8 20))))
                  (COND((EQUAL ALNNR 999) (THROW 'ENDMARK NIL))
                        ((NOT (EQUAL ALNNR 888)) (EXIT NIL))
                        (T NIL)))
            (LETS  ((FRAME (@FINSTANTIATE 'AIRLINE-LIST 'AIRLINE)))
             (@FPUT FRAME 'NUMBER '¥VALUE ALNNR)
             (@FPUT FRAME 'ABBREV '¥VALUE ALN)
             (@FPUT FRAME 'NAME '¥VALUE ALNNM))))
    (@FDESTROY 'AIRLINE)
    (US:POSTPROCESS-READ-LIST 'AIRLINE-LIST))

(DEFUN US:READ-TYPE-LIST  NIL
    (SETQ CTNR 0)
    (CATCH  'ENDMARK
       (LOOP(LOOP (SETQ ATFBF (SUBATF "DDD"))
                  (SETQ  TYP
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 0 3)))
                         TYPCD
                         (US:STRING-TO-NUMBER
                          (SUBSTRING ATFBF 4 6))
                         TYPNM
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 7 17))))
                  (COND((EQUAL TYP (INTERN "999"))
                        (THROW 'ENDMARK NIL))
                       ((NOT (EQUAL TYP (INTERN "888")))
                        (EXIT NIL))
                       (T NIL)))
            (LETS  ((FRAME (@FINSTANTIATE 'TYPE-LIST 'TYPE)))
             (@FPUT FRAME 'NUMBER '¥VALUE (INCR CTNR 1))
             (@FPUT FRAME 'ABBREV '¥VALUE TYP)
             (@FPUT FRAME 'CODE '¥VALUE TYPCD)
             (@FPUT FRAME 'NAME '¥VALUE TYPNM))))
    (@FDESTROY 'TYPE)
    (US:POSTPROCESS-READ-LIST 'TYPE-LIST))

(DEFUN US:READ-FLIGHT-SCHEDULE-LIST  NIL
    (SETQ ATFBF (SUBATF "DDD"))
    (FORMAT "THE VALUE OF ATFBUFF IS /C. /N" ATFBF)
    (SETQ  YYYY  (US:STRING-TO-NUMBER (SUBSTRING ATFBF 0 4))
           MM  (US:STRING-TO-NUMBER (SUBSTRING ATFBF 5 7))
           DD  (US:STRING-TO-NUMBER (SUBSTRING ATFBF 8 10)))
    (LETS  ((FRAME (@FCREATE 'YEAR-MONTH-DATE)))
     (@FPUT FRAME 'CLASS '¥VALUE 'INDIVIDUAL)
     (@FPUT FRAME 'YEAR '¥VALUE YYYY)
     (@FPUT FRAME 'MONTH '¥VALUE MM)
     (@FPUT FRAME 'DATE '¥VALUE DD))
    (SETQ CTNR 0)
    (CATCH  'ENDMARK
       (LOOP(SETQ ATFBF (SUBATF "DDD"))
            (SETQ  LDEP
                   (INTERN (SUBSTRING ATFBF 0 3))
                   LARR
                   (INTERN (SUBSTRING ATFBF 7 10)))
            (COND((EQUAL LDEP (INTERN "999")) (THROW 'ENDMARK NIL))
                  (T NIL))
            (LOOP(SETQ ATFBF (SUBATF "DDD"))
                 (SETQ  FALN
                        (INTERN (SUBSTRING ATFBF 0 3))
                        FFLT
                        (US:STRING-TO-NUMBER
                         (SUBSTRING ATFBF 4 7))
                        FTYP
                        (INTERN
                          (US:REMOVE-EXTRA-SPACE
                            (SUBSTRING ATFBF 8 11)))
                        FDTM
                        (US:STRING-TO-NUMBER
                         (SUBSTRING ATFBF 12 16))
                        FATM
                        (US:STRING-TO-NUMBER
                         (SUBSTRING ATFBF 17 21)))
                 (COND((EQUAL FALN (INTERN "888")) (EXIT NIL))
                       (T NIL))
                 (LETS
                   ((FRAME
                        (@FINSTANTIATE 'FLIGHT-SCHEDULE-LIST
                          'FLIGHT-SCHEDULE)))
                  (@FPUT FRAME 'NUMBER '¥VALUE (INCR CTNR 1))
                  (@FPUT FRAME 'DEPARTURE-AIRPORT '¥VALUE LDEP)
                  (@FPUT FRAME 'ARRIVAL-AIRPORT '¥VALUE LARR)
                  (@FPUT FRAME 'AIRLINE '¥VALUE FALN)
                  (@FPUT FRAME 'FLIGHT-NUMBER '¥VALUE FFLT)
                  (@FPUT FRAME 'TYPE '¥VALUE FTYP)
                  (@FPUT FRAME 'DEPARTURE-TIME '¥VALUE FDTM)
```

付図２－４　プロダクション型知識ベース(4)

```
                    (@FPUT FRAME 'ARRIVAL-TIME '¥VALUE FATM)))))
    (@FDESTROY 'FLIGHT-SCHEDULE)
    (US:POSTPROCESS-READ-LIST 'FLIGHT-SCHEDULE-LIST))

(DEFUN US:READ-IDLE-LINE-LIST  NIL
    (SETQ CTNR 0)
    (CATCH  'ENDMARK
       (LOOP(LOOP (SETQ ATFBF (SUBATF "DDD"))
                  (SETQ  NONDEP
                         (INTERN (SUBSTRING ATFBF 0 3))
                         NONARR
                         (INTERN (SUBSTRING ATFBF 7 10)))
                  (COND((EQUAL NONDEP (INTERN "999"))
                        (THROW 'ENDMARK NIL))
                       ((NOT (EQUAL NONDEP (INTERN "888")))
                        (EXIT NIL))
                       (T NIL)))
            (LETS  ( (FRAME
                       (@FINSTANTIATE 'IDLE-LINE-LIST 'IDLE-LINE)))
             (@FPUT FRAME 'NUMBER '¥VALUE (INCR CTNR 1))
             (@FPUT FRAME 'DEPARTURE-AIRPORT '¥VALUE NONDEP)
             (@FPUT FRAME 'ARRIVAL-AIRPROT '¥VALUE NONARR))))
    (@FDESTROY 'IDLE-LINE)
    (US:POSTPROCESS-READ-LIST 'IDLE-LINE-LIST))

(DEFUN US:READ-FIX-LIST  NIL
    (CATCH  'ENDMARK
       (LOOP(LOOP (SETQ ATFBF (SUBATF "DDD"))
                  (SETQ  FIXNR
                         (DIFFERENCE
                           (US:STRING-TO-NUMBER
                             (SUBSTRING ATFBF 0 3))
                          100)
                         FIX
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 4 7)))
                         FIXNM
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 8 24))))
                  (COND((EQUAL FIXNR 899) (THROW 'ENDMARK NIL))
                       ((NOT (EQUAL FIXNR 788)) (EXIT NIL))
                       (T NIL)))
            (LETS  ((FRAME (@FINSTANTIATE 'FIX-LIST 'FIX)))
             (@FPUT FRAME 'NUMBER '¥VALUE FIXNR)
             (@FPUT FRAME 'ABBREV '¥VALUE FIX)
             (@FPUT FRAME 'NAME '¥VALUE FIXNM))))
    (@FDESTROY 'FIX)
    (US:POSTPROCESS-READ-LIST 'FIX-LIST))

(DEFUN US:READ-SEGMENT-LIST  NIL
    (SETQ CTNR 0)
    (CATCH  'ENDMARK
       (LOOP(LOOP (SETQ ATFBF (SUBATF "DDD"))
                  (SETQ  SEGBGN
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 0 3)))
                         SEGEND
                         (INTERN
                           (US:REMOVE-EXTRA-SPACE
                             (SUBSTRING ATFBF 4 7)))
                         SEGDIST
                         (US:STRING-TO-NUMBER
                          (SUBSTRING ATFBF 8 11)))
                  (COND((EQUAL SEGBGN (INTERN "999"))
                        (THROW 'ENDMARK NIL))
                       ((NOT (EQUAL SEGBGN (INTERN "888")))
                        (EXIT NIL))
                       (T NIL)))
            (LETS  ((FRAME (@FINSTANTIATE 'SEGMENT-LIST 'SEGMENT)))
             (@FPUT FRAME 'NUMBER '¥VALUE (INCR CTNR 1))
             (@FPUT FRAME 'BEGIN-FIX '¥VALUE SEGBGN)
             (@FPUT FRAME 'END-FIX '¥VALUE SEGEND)
             (@FPUT FRAME 'DISTANCE '¥VALUE SEGDIST))))
    (@FDESTROY 'SEGMENT)
    (US:POSTPROCESS-READ-LIST 'SEGMENT-LIST))

(DEFUN US:READ-ROUTE-LIST  NIL
    (SETQ CTNR 0)
    (CATCH  'ENDMARK
       (LOOP(LOOP (SETQ ATFBF1 (SUBATF "DDD") ATFBF2 (SUBATF "DDD"))
                  (SETQ  RDEP
                         (INTERN (SUBSTRING ATFBF1 0 3))
                         RARR
                         (INTERN (SUBSTRING ATFBF1 4 7))
                         RDIC
                         (INTERN (SUBSTRING ATFBF1 8 9))
                         RFIX1
                         (SUBSTRING ATFBF1 10 57)
                         RFIX2
                         (SUBSTRING ATFBF2 10 57))
                  (COND((EQUAL RDEP (INTERN "999"))
                        (THROW 'ENDMARK NIL))
                       ((NOT (EQUAL RDEP (INTERN "888")))
                        (EXIT NIL))
                       (T NIL)))
            (LETS  ((FRAME (@FINSTANTIATE 'ROUTE-LIST 'ROUTE)))
             (@FPUT FRAME 'NUMBER '¥VALUE (INCR CTNR 1))
             (@FPUT FRAME 'DEPARTURE-AIRPORT '¥VALUE RDEP)
```

付図２－５　プロダクション型知識ベース(5)

```
              (@FPUT FRAME 'ARRIVAL-AIRPORT '¥VALUE RARR)
              (@FPUT FRAME 'DIRECTION '¥VALUE RDIC)
              (@FPUTFRAME
                     'PASSING-FIX
                     '¥VALUE
                     (US:LISTING-OF-PASSING-FIX RFIX1 RFIX2)))))
    (@FDESTROY 'ROUTE)
    (US:POSTPROCESS-READ-LIST 'ROUTE-LIST))

(DEFUN US:POSTPROCESS NIL (FORMAT "/N SEARCHING HAS FINISHED /N"))

(DEFUN US:STRING-TO-NUMBER  (STR)
    (LETS  ((POS (STRING-SEARCH-CHAR "." STR)) (FCHR (SREF STR 0))
           (BPOS (COND ( (OR (EQUAL FCHR (CHARACTER "-"))
                             (EQUAL FCHR (CHARACTER "+")))
                       1)
                      (T 0)))
           (SIGN (COND ((EQUAL FCHR (CHARACTER "-")) -1) (T 1))))
     (TIMESSIGN
           (COND(POS (PLUS (US:STR-TO-FIX (SUBSTRING STR BPOS POS))
                           (US:STR-TO-DEC (SUBSTRING STR (ADD1 POS)))))
                (T (US:STR-TO-FIX (SUBSTRING STR BPOS)))))))

(DEFUN US:STR-TO-FIX  (STR)
    (DO((ZERO (CHARACTER "0"))
        (NUMO
           (PLUS (TIMES NUM 10) (DIFFERENCE (SREF STR POS) ZERO)))
        (POS 0 (ADD1 POS)))
       ((EQUAL POS (STRING-LENGTH STR)) NUM)))

(DEFUN US:STR-TO-DEC  (STR)
    (DO((ZERO (CHARACTER "0"))
        (NUMO.0
           (PLUS(QUOTIENT NUM 10)
                (DIFFERENCE (SREF STR POS) ZERO)))
        (POS (SUB1 (STRING-LENGTH STR)) (SUB1 POS)))
       ((EQUAL POS -1) (QUOTIENT NUM 10))))

(DEFUN US:REMOVE-EXTRA-SPACE  (STR)
    (LETS  ((STRL (STRING-LENGTH STR)))
     (COND((EQUAL STRL 1) STR)
          (
           (EQUAL (SUBSTRING STR (SUB1 STRL)) " ")
           (US:REMOVE-EXTRA-SPACE (SUBSTRING STR 0 (SUB1 STRL))))
          (T STR))))

(DEFUN US:POSTPROCESS-READ-LIST  (PNAME)
    (MAPCAR  (@FGET PNAME 'INSTANCE '¥LINK)
       (FUNCTION (LAMBDA (X) (US:CONVERT-FNAME X PNAME)))))

(DEFUN US:CONVERT-FNAME  (FNAME Y)
    (LETS  ( (NEW-FNAME
                (LETS  ((SUFIX (@FGET FNAME 'NUMBER '¥VALUE '(0)))
                        (SUFIX1 (REMAINDER (QUOTIENT SUFIX 1) 10))
                        (SUFIX2 (REMAINDER (QUOTIENT SUFIX 10) 10))
                        (SUFIX3 (REMAINDER (QUOTIENT SUFIX 100) 10))
                        (SUFIX4
                           (REMAINDER (QUOTIENT SUFIX 1000) 10)))
                 (STRING-APPEND
                    (SUBSTRING
                       FNAME
                       0
                       (DIFFERENCE (STRING-LENGTH FNAME) 4))
                    (SUBSTRING "0123456789" SUFIX4 (ADD1 SUFIX4))
                    (SUBSTRING "0123456789" SUFIX3 (ADD1 SUFIX3))
                    (SUBSTRING "0123456789" SUFIX2 (ADD1 SUFIX2))
                    (SUBSTRING "0123456789" SUFIX1
                     (ADD1 SUFIX1))))))
     (@FREMOVE Y 'INSTANCE '¥LINK FNAME)
     (@FPUT Y 'INSTANCE '¥LINK (INTERN NEW-FNAME))
     (@FCOPY FNAME (INTERN NEW-FNAME))
     (@FDESTROY FNAME)))

(DEFUN US:LISTING-OF-PASSING-FIX  (STR1 STR2)
    (LETS  ((LIST1 (US:CUTOUT-FROM-STRING STR1))
            (LIST2 (US:CUTOUT-FROM-STRING STR2)))
     (US:STRING-TO-SYMBOL-IN-LIST
        (US:REMOVE-EXTRA-FIX (APPEND LIST1 LIST2)))))

(DEFUN US:CUTOUT-FROM-STRING  (STR)
    (DO((POS 0 (PLUS POS 4))
        (LIST1 NIL (CONS (SUBSTRING STR POS (PLUS POS 3)) LIST1)))
       ((EQUAL POS 48) (REVERSE LIST1))))

(DEFUN US:STRING-TO-SYMBOL-IN-LIST  (LIST)
    (MAPCAR  LIST
       (FUNCTION (LAMBDA (X) (INTERN (US:REMOVE-EXTRA-SPACE X))))))

(DEFUN US:REMOVE-EXTRA-FIX  (LIST)
    (LETS  ((LISTR (REVERSE LIST)))
     (REVERSE (US:REMOVE-EXTRA-FIX1 LISTR))))

(DEFUN US:REMOVE-EXTRA-FIX1  (LISTR)
    (COND((NOT (EQUAL (CAR LISTR) "   ")) LISTR)
          (T (US:REMOVE-EXTRA-FIX1 (CDR LISTR)))))

(DEFUN US:INITIALIZE-RANDOM-NUMBER-GENERATOR  NIL
    (SETQ RANIND 1 RANSKP 10))

(DEFUN US:ORDERING-FLIGHT-SCHEDULE  NIL
    (LETS  ( (FTABLE (MAPCAR
```

付図２－６　プロダクション型知識ベース(6)

```
                        (@FGET 'FLIGHT-SCHEDULE-LIST 'INSTANCE '¥LINK)
                         (FUNCTION
                            (LAMBDA (X)
                               (LIST
                                 (@FGET X 'DEPARTURE-TIME '¥VALUE '(O))
                                 X))))))
      (US:ORDERING
         (SORTFTABLE
               (FUNCTION (LAMBDA (U V) (< (CAR U) (CAR V))))))))

(DEFUN US:ORDERING  (OTABLE)
    (DO((CNTR 1 (PLUS CNTR 1)) (WORK OTABLE (CDR WORK)))
        ((NULL WORK) NIL)
        (@FPUT (CADAR WORK) 'ORDER '¥VALUE CNTR)))

(DEFUN US:ROUTE?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD)))
            (FLDEP (@FGET FLSCH 'DEPARTURE-AIRPORT '¥VALUE '(O)))
            (FLARR (@FGET FLSCH 'ARRIVAL-AIRPORT '¥VALUE '(O)))
            (RLGTH (US:ROUTE-LENGTH FLDEP FLARR))
            (RRANK (US:ROUTE-RANK RLGTH)))
     (SETQ NOW-ROUTE (LIST FLDEP 'TO FLARR RRANK RLGTH))
     NOW-ROUTE))

(DEFUN US:ROUTE-LENGTH  (RD RA)
    (LETS  ((SET1 (@FRAME-FIND 'DEPARTURE-AIRPORT RD))
            (SET2 (@FRAME-FIND 'ARRIVAL-AIRPORT RA))
            (SET3 (@FGET 'ROUTE-LIST 'INSTANCE '¥LINK))
            (ROUT (CAR
                     (US:INTERSECTION (US:INTERSECTION SET1 SET3)
                        (US:INTERSECTION SET2 SET3)))))
     (US:ROUTE-LENGTH1 ROUT)))

(DEFUN US:ROUTE-LENGTH1  (RT)
    (LETS  ((FLIST (@FGET RT 'PASSING-FIX '¥VALUE '(O))))
     (US:ROUTE-LENGTH2 FLIST)))

(DEFUN US:ROUTE-LENGTH2  (FLIST)
    (COND((EQ (LENGTH FLIST) 1) 0)
          (T(/+ (@FGET (CAR
                          (US:UNION
                             (US:INTERSECTION
                               (@FRAME-FIND 'BEGIN-FIX (CAR FLIST))
                               (@FRAME-FIND 'END-FIX (CADR FLIST)))

                             (US:INTERSECTION
                               (@FRAME-FIND 'BEGIN-FIX (CADR FLIST))
                               (@FRAME-FIND 'END-FIX (CAR FLIST)))))
                        'DISTANCE
                        '¥VALUE
                        '(O))
                (US:ROUTE-LENGTH2 (CDR FLIST))))))

(DEFUN US:ROUTE-RANK  (RL)
    (COND((<= RL 200) 'SHORT)
          ((AND (>= RL 201) (<= RL 400)) 'MEDIUM)
          ((>= RL 401) 'LONG)
          (T (FORMAT "ROUTE-RANK ERROR"))))

(DEFUN US:UNION  (X Y)
    (COND((NULL X) Y)
          ((MEMBER (CAR X) Y) (US:UNION (CDR X) Y))
          (T (US:UNION (CDR X) (CONS (CAR X) Y)))))

(DEFUN US:INTERSECTION  (X Y)
    (LETS  ((XUY (US:UNION X Y)))
     (MAPCAN  XUY
        (FUNCTION
         (LAMBDA  (E)
          (COND((AND (MEMBER E X) (MEMBER E Y)) (LIST E))
                (T NIL)))))))

(DEFUN US:AIRLINE?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD))))
     (SETQ NOW-AIRLINE (@FGET FLSCH 'AIRLINE '¥VALUE '(O)))
     NOW-AIRLINE))

(DEFUN US:FLIGHT-NUMBER?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD))))
     (SETQ  NOW-FLIGHT-NUMBER
            (@FGET FLSCH 'FLIGHT-NUMBER '¥VALUE '(O)))
     NOW-FLIGHT-NUMBER))

(DEFUN US:TYPE?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD)))
            (TYP (@FGET FLSCH 'TYPE '¥VALUE '(O)))
             (TYPE (CAR (US:INTERSECTION (@FRAME-FIND 'ABBREV TYP)
                          (@FGET 'TYPE-LIST 'INSTANCE '¥LINK))))
            (TYPCD (@FGET TYPE 'CODE '¥VALUE '(O))))
     (SETQ  NOW-TYPE  (LIST TYP
                        (LETS ((RK (// TYPCD 10)))
                           (COND ((EQ RK 1) 'TYPE1)
                             ((EQ RK 2) 'TYPE2) ((EQ RK 3) 'TYPE3)
                             (T (FORMAT "TYPE-RANK ERROR"))))))
     NOW-TYPE))

(DEFUN US:DEPARTURE-AIRPORT?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD))))
     (SETQ  NOW-DEPARTURE-AIRPORT
            (@FGET FLSCH 'DEPARTURE-AIRPORT '¥VALUE '(O)))
     NOW-DEPARTURE-AIRPORT))
```

付図２－７　プロダクション型知識ベース(7)

```
(DEFUN US:ARRIVAL-AIRPORT?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD))))
     (SETQ  NOW-ARRIVAL-AIRPORT
            (@FGET FLSCH 'ARRIVAL-AIRPORT '¥VALUE '(0)))
     NOW-ARRIVAL-AIRPORT))

(DEFUN US:DEPARTURE-TIME?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD))))
     (SETQ  NOW-DEPARTURE-TIME
            (@FGET FLSCH 'DEPARTURE-TIME '¥VALUE '(0)))
     NOW-DEPARTURE-TIME))

(DEFUN US:ARRIVAL-TIME?  (ORD)
    (LETS  ((FLSCH (CAR (@FRAME-FIND 'ORDER ORD))))
     (SETQ NOW-ARRIVAL-TIME (@FGET FLSCH 'ARRIVAL-TIME '¥VALUE '(0)))
     NOW-ARRIVAL-TIME))

(DEFUN US:SPEED?  NIL
    (LETS  ((SCHTM (US:FLIGHT-TIME-IN-SCHEDULE))
            (GDSVT (US:GROUND-SERVICE-TIME-TOTAL))
            (FDIST (US:FLIGHT-DISTANCE))
             (SPEED (COND
                        ((>=¥ GDSVT (FLOAT SCHTM))
                           (FORMAT
                              "CAN NOT CALCULATE SPEED FOR /C TO /C /C /C"
                             NOW-DEPARTURE-AIRPORT
                             NOW-ARRIVAL-AIRPORT NOW-AIRLINE
                             NOW-FLIGHT-NUMBER))

                        (T
                           (/+¥
                              (//¥ (*¥ 60.0 (FLOAT FDIST))
                                   (/-¥ (FLOAT SCHTM) GDSVT))
                             0.5)))))
     (SETQ NOW-SPEED (* (// (/+ (FIX SPEED) 5) 10) 10))
     NOW-SPEED))

(DEFUN US:FLIGHT-TIME-IN-SCHEDULE  NIL
    (LETS  ((DTM NOW-DEPARTURE-TIME) (ATM NOW-ARRIVAL-TIME)
            (MIN1 (US:HM-TO-MIN DTM)) (MIN2 (US:HM-TO-MIN ATM)))
     (/- MIN2 MIN1)))

(DEFUN US:GROUND-SERVICE-TIME-TOTAL  NIL
    (LETS  ((IND (US:INDEX-FOR-GDSVTM-TOTAL))
            (GST (@VALUE '@DATA 'GROUND-SERVICE-TIME-MEANS)))
     (DO((TTL 0.0 (/+¥ TTL (NTH (/- IND 1) (CAR MTX))))
         (MTX GST (CDR MTX)))
        ((NULL MTX) TTL))))

(DEFUN US:INDEX-FOR-GDSVTM-TOTAL  NIL
    (LETS  ((TYPE NOW-TYPE) (TABR (CAR TYPE)) (TRNK (CADR TYPE)))
     (COND((MEMBER TABR '(D41 M81 M87)) 1)
           (T (US:STRING-TO-NUMBER (SUBSTRING TRNK 4 5))))))

(DEFUN US:FLIGHT-DISTANCE NIL (FIFTH NOW-ROUTE))

(DEFUN US:HM-TO-MIN  (HHMM)
    (/+ (* (// HHMM 100) 60) (REMAINDER HHMM 100)))

(DEFUN US:MIN-TO-HM  (MIN)
    (/+ (* (// MIN 60) 100) (REMAINDER MIN 60)))

(DEFUN US:ALTITUDE?  NIL
    (LETS  ((LVS (US:POSSIBLE-LEVELS)))
     (FORMAT "SELECT FLIGHT-LEVEL OF FIRST-FLIGHT FROM /C /N" LVS)
     (LOOP(LETS  ((LV (RIND "LEVEL=>")))
            (AND (MEMQ LV LVS) (SETQ NOW-ALTITUDE LV) (EXIT LV)))))
    (US:CONFIRM-ALTITUDE))

(DEFUN US:POSSIBLE-LEVELS  NIL
    (LETS  ((TR (US:TYPE-RANK1)) (RR (US:ROUTE-RANK1))
            (RD (US:ROUTE-DIRECTION1))
            (FLS (@VALUE '@DATA 'FLIGHT-LEVELS)))
     (US:SELECT-POSSIBLE-LEVELS TR RR RD FLS)))

(DEFUN US:TYPE-RANK1 NIL (CADR NOW-TYPE))

(DEFUN US:ROUTE-RANK1 NIL (FOURTH NOW-ROUTE))

(DEFUN US:ROUTE-DIRECTION1  NIL
    (LETS  ((RDEP (CAR NOW-ROUTE)) (RARR (CADDR NOW-ROUTE))
            (SET1 (@FRAME-FIND 'DEPARTURE-AIRPORT RDEP))
            (SET2 (@FRAME-FIND 'ARRIVAL-AIRPORT RARR))
            (SET3 (@FGET 'ROUTE-LIST 'INSTANCE '¥LINK))
             (ROUT (CAR
                       (US:INTERSECTION (US:INTERSECTION SET1 SET3)
                         (US:INTERSECTION SET2 SET3)))))
     (@FGET ROUT 'DIRECTION '¥VALUE '(0))))

(DEFUN US:SELECT-POSSIBLE-LEVELS  (X Y Z W)
    (LETS  ( (W1 (COND ((EQUAL X 'TYPE1) (CAR W))
                       ((EQUAL X 'TYPE2) (CADR W))
                       ((EQUAL X 'TYPE3) (CADDR W))
                       (T (FORMAT "TYPE-RANK ERROR"))))

              (W2 (COND ((EQUAL Y 'SHORT) (CAR W1))
                        ((EQUAL Y 'MEDIUM) (CADR W1))
                        ((EQUAL Y 'LONG) (CADDR W1))
                        (T (FORMAT "ROUTE-RANK ERROR"))))

              (W3 (COND ((EQUAL Z 'E) (CAR W2))
```

付図２－８　プロダクション型知識ベース(8)

```
                        ((EQUAL Z 'W) (CADR W2))
                        (T (FORMAT "ROUTE-DIRECTION ERROR")))))
     W3))

(DEFUN US:CONFIRM-ALTITUDE  NIL
    (LETS  ((ROUT NOW-ROUTE) (TYPE NOW-TYPE) (ALTT NOW-ALTITUDE)
            (SPED NOW-SPEED) (FLTM (US:FLIGHT-TIME ROUT SPED))
            (ALTS (US:ALTITUDES ALTT)))
     (DO((WORK ALTS (CDR WORK)))
         (
          (>= (US:LEVEL-FLIGHT-TIME FLTM TYPE (CAR WORK)) 0)
          (SETQ NOW-CONFIRMED-ALTITUDE (CAR WORK))
          NOW-CONFIRMED-ALTITUDE))))

(DEFUN US:FLIGHT-TIME  (ROUTE SPEED)
    (LETS  ((SET1 (@FRAME-FIND 'DEPARTURE-AIRPORT (NTH 0 ROUTE)))
            (SET2 (@FRAME-FIND 'ARRIVAL-AIRPORT (NTH 2 ROUTE)))
            (SET3 (@FGET 'ROUTE-LIST 'INSTANCE '¥LINK))
            (ROUT (CAR (US:INTERSECTION
                        (US:INTERSECTION SET1 SET3)
                        (US:INTERSECTION SET2 SET3))))
            (FIXS (@FGET ROUT 'PASSING-FIX '¥VALUE '(0))))
     (US:FLIGHT-TIME1 FIXS SPEED)))

(DEFUN US:FLIGHT-TIME1  (FIXS SPED)
    (COND((EQ (LENGTH FIXS) 1) 0)
          (T(/+ (LETS  ( (DIST
                           (@FGET
                              (CAR
                                 (US:UNION
                                    (US:INTERSECTION
                                       (@FRAME-FIND 'BEGIN-FIX
                                         (CAR FIXS))

                                       (@FRAME-FIND 'END-FIX
                                         (CADR FIXS)))

                                    (US:INTERSECTION
                                       (@FRAME-FIND 'BEGIN-FIX
                                         (CADR FIXS))

                                       (@FRAME-FIND 'END-FIX
                                         (CAR FIXS)))))
                                'DISTANCE '¥VALUE '(0))))
                  (FIX(/+¥ (*¥ 60.0 (//¥ (FLOAT DIST) (FLOAT SPED)))
                           0.5)))
                (US:FLIGHT-TIME1 (CDR FIXS) SPED)))))

(DEFUN US:ALTITUDES  (ALTT)
    (LETS  ((LVS (US:POSSIBLE-LEVELS)))
     (MEMBER ALTT (REVERSE (PUSH 0 LVS)))))

(DEFUN US:LEVEL-FLIGHT-TIME  (FLTM TYPE ALTC)
    (LETS  ((RATE (COND ((EQUAL (CADR TYPE) 'TYPE1) 10) (T 20)))
            (CLDS (/+¥ (//¥ (FLOAT ALTC) (FLOAT RATE)) 0.5))
            (CDTM (FIX CLDS)))
     (/- FLTM (* 2 CDTM))))

(DEFUN US:TOWING?  (DD)
    (LETS  ((TYP NOW-TYPE) (BGN (US:TOWING-BEGIN-TIME DD))
            (END (US:TOWING-END-TIME BGN TYP)))
     (SETQ NOW-TOWING (LIST BGN END))
     NOW-TOWING))

(DEFUN US:TOWING-BEGIN-TIME  (DD)
    (US:MIN-TO-HM (/+ (US:HM-TO-MIN NOW-DEPARTURE-TIME) DD)))

(DEFUN US:TOWING-END-TIME  (BGN TYP)
    (US:MIN-TO-HM (/+ (US:HM-TO-MIN BGN) (US:TOWING-TIME TYP))))

(DEFUN US:TOWING-TIME  (TYPE)
    (LETS  ((TYP (CAR TYPE)) (TRK (CADR TYPE))
            (IND (COND ((MEMBER TYP '(D41 M81 M87)) 1)
                       (T
                        (US:STRING-TO-NUMBER (SUBSTRING TRK 4 5)))))
            (MTX (@VALUE '@DATA 'TOWING-TIME-MATRIX))
            (DNF (COND ((= IND 1) (CAR MTX))
                       ((= IND 2) (CADR MTX))
                       ((= IND 3) (CADDR MTX))
                       (T (FORMAT "TOWING-TIME ERROR")))))
     (US:TIME-DECISION-BY-DNF-&-RN DNF)))

(DEFUN US:TIME-DECISION-BY-DNF-&-RN  (DNF)
    (LETS  ((RANNO (SUBRAN RANIND RANSKP)))
     (DO((N 0 (/+ N 1)))
         (
          (< RANNO (CAR (NTH N DNF)))
          (SETQ RANIND 2)
          (CADR (NTH N DNF))))))

(DEFUN US:TAXIING-BEFORE-TAKE-OFF?  NIL
    (LETS  ((TYP NOW-TYPE) (BGN (CADR NOW-TOWING))
            (END (US:TAKE-OFF-TAXI-END-TIME BGN TYP)))
     (SETQ NOW-TAXIING-BEFORE-TAKE-OFF (LIST BGN END))
     NOW-TAXIING-BEFORE-TAKE-OFF))

(DEFUN US:TAKE-OFF-TAXI-END-TIME  (BGN TYP)
    (US:MIN-TO-HM
       (/+ (US:HM-TO-MIN BGN) (US:TAKE-OFF-TAXI-TIME TYP))))

(DEFUN US:TAKE-OFF-TAXI-TIME  (TYP)
```

付図２－９　プロダクション型知識ベース(9)

```
    (LETS  ((TRK (CADR TYP))
            (IND (US:STRING-TO-NUMBER (SUBSTRING TRK 4 5)))
            (MTX (@VALUE '@DATA 'TAKE-OFF-TAXI-TIME-MATRIX))
             (DNF (COND ((= IND 1) (CAR MTX))
                        ((= IND 2) (CADR MTX))
                        ((= IND 3) (CADDR MTX))
                        (T (FORMAT "TAKE-OFF-TAXI-TIME ERROR")))))
     (US:TIME-DECISION-BY-DNF-&-RN DNF)))

(DEFUN US:RUNWAY-OCCUPANCY-FOR-TAKE-OFF?  (LAST-CLEARANCE)
    (LETS  ((CAND (US:OCCUPANCY-CANDIDATE-T))
            (BLKD (US:BLOCKED-INTERVALS-T LAST-CLEARANCE)))
     (SETQ  NOW-RUNWAY-OCCUPANCY-FOR-TAKE-OFF
            (COND((NULL LAST-CLEARANCE) CAND)
                 (T (US:POSSIBLE-OCCUPANCY CAND BLKD))))
     NOW-RUNWAY-OCCUPANCY-FOR-TAKE-OFF))

(DEFUN US:OCCUPANCY-CANDIDATE-T  NIL
    (LETS  ((TYP NOW-TYPE) (BGN (CADR NOW-TAXIING-BEFORE-TAKE-OFF))
            (END (US:TAKE-OFF-RUNWAY-OCCUPANCY-END-TIME BGN TYP)))
     (LIST BGN END)))

(DEFUN US:TAKE-OFF-RUNWAY-OCCUPANCY-END-TIME  (BGN TYP)
    (US:MIN-TO-HM
       (/+(US:HM-TO-MIN BGN)
          (US:TAKE-OFF-RUNWAY-OCCUPANCY-TIME TYP))))

(DEFUN US:TAKE-OFF-RUNWAY-OCCUPANCY-TIME  (TYP)
    (LETS  ((TRK (CADR TYP))
            (IND (US:STRING-TO-NUMBER (SUBSTRING TRK 4 5)))
            (MTX (@VALUE '@DATA 'TAKE-OFF-RUNWAY-TIME-MATRIX))
             (DNF (COND ((= IND 1) (CAR MTX))
                        ((= IND 2) (CADR MTX))
                        ((= IND 3) (CADDR MTX))
                        (T
                          (FORMAT
                             "TAKE-OFF-RUNWAY-OCCUPANCY-TIME ERROR")))))
     (US:TIME-DECISION-BY-DNF-&-RN DNF)))

(DEFUN US:BLOCKED-INTERVALS-T  (LAST-CLEARANCE)
    (LETS  ((APOR NOW-DEPARTURE-AIRPORT)
            (SET1 (DO ((LINK LAST-CLEARANCE (@VALUE LINK 'LAST))
                       (CHAIN NIL (PUSH LINK CHAIN)))
                      ((NULL LINK) CHAIN)))

            (SET2 (@FIND-IN-LEVEL
                      'ATC-CLEARANCE
                      (LIST (LIST 'DEPARTURE-AIRPORT APOR))))

            (SET3 (@FIND-IN-LEVEL
                      'ATC-CLEARANCE
                      (LIST (LIST 'ARRIVAL-AIRPORT APOR))))
           (SET4 (US:INTERSECTION SET1 SET2))
           (SET5 (US:INTERSECTION SET1 SET3))
            (DBLK (COND ((NULL SET4) NIL)
                        (T
                          (MAPCAR SET4
                              (FUNCTION
                                 (LAMBDA (X)
                                    (@VALUE X
                                       'RUNWAY-OCCUPANCY-FOR-TAKE-OFF)))))))

            (ABLK (COND ((NULL SET5) NIL)
                        (T
                          (MAPCAR SET5
                              (FUNCTION
                                 (LAMBDA (X)
                                    (@VALUE X
                                       'RUNWAY-OCCUPANCY-FOR-LANDING)))))))
            (BLKD (US:UNION DBLK ABLK)))
     (COND((NULL BLKD) NIL)
           (T(SORT BLKD
                   (FUNCTION
                    (LAMBDA (U V) (< (CAR U) (CAR V)))))))))

(DEFUN US:POSSIBLE-OCCUPANCY  (CAND BLKD)
    (DO((WORK1 CAND
               (US:NEW-CADIDATE
                  WORK1
                  (CAR(SOME WORK2
                            (FUNCTION
                               (LAMBDA (X)
                                  (NOT (US:OVERLAP-TEST WORK1 X))))))))
        (WORK2BLKD
               (CDR(SOME WORK2
                         (FUNCTION
                            (LAMBDA (X)
                              (NOT (US:OVERLAP-TEST WORK1 X))))))))
       (
        (OR(NULL WORK2)
           (EVERY  WORK2
              (FUNCTION (LAMBDA (X) (US:OVERLAP-TEST WORK1 X)))))
        WORK1)))

(DEFUN US:OVERLAP-TEST  (INT1 INT2)
    (LETS  ((T1 (US:HM-TO-MIN (CAR INT1)))
            (T2 (US:HM-TO-MIN (CADR INT1)))
            (T3 (US:HM-TO-MIN (CAR INT2)))
            (T4 (US:HM-TO-MIN (CADR INT2))))
     (COND ((OR (<= T4 T1) (>= T3 T2)) T) (T NIL))))
```

付図２—10　プロダクション型知識ベース(10)

```
(DEFUN US:NEW-CADIDATE  (INT1 INT2)
    (LETS  ((T1 (US:HM-TO-MIN (CAR INT1)))
            (T2 (US:HM-TO-MIN (CADR INT1)))
            (T4 (US:HM-TO-MIN (CADR INT2))) (TIME (/- T2 T1)))
     (LIST (US:MIN-TO-HM T4) (US:MIN-TO-HM (/+ T4 TIME)))))

(DEFUN US:FLIGHT-ON-EACH-SEGMENT?  NIL
    (LETS  ((ROUT NOW-ROUTE) (TYPE NOW-TYPE)
            (ALTD NOW-CONFIRMED-ALTITUDE) (SPED NOW-SPEED)
            (FLTM (US:FLIGHT-TIME ROUT SPED)))
     (US:MAKE-FLIGHT-PATTERN FLTM TYPE ALTD)
     (SETQ  NOW-FLIGHT-ON-EACH-SEGMENT
            (US:MAPPING-PATTERN-ON-ROUTE ROUT SPED))
     NOW-FLIGHT-ON-EACH-SEGMENT))

(DEFUN US:MAKE-FLIGHT-PATTERN  (FLTM TYPE ALTD)
    (SETQ  FPATTERN  (LETS
                       ((LVTM (US:LEVEL-FLIGHT-TIME FLTM TYPE ALTD)))
                        (COND
                           ((AND (/0< LVTM) (/0< ALTD))
                             (US:FLIGHT-PATTERN1 FLTM LVTM ALTD))

                           ((AND (/0= LVTM) (/0< ALTD))
                             (US:FLIGHT-PATTERN2 FLTM ALTD))
                          ((/0= ALTD) (US:FLIGHT-PATTERN3 FLTM))
                          (T (FORMAT "FLIGHT-PATTERN ERROR"))))))

(DEFUN US:FLIGHT-PATTERN1  (FLTM LVTM ALTD)
    (LETS  ((CDTM (// (/- FLTM LVTM) 2)))
     (LIST(LIST 0 0)
          (LIST CDTM ALTD)
          (LIST (/+ CDTM LVTM) ALTD)
          (LIST FLTM 0))))

(DEFUN US:FLIGHT-PATTERN2  (FLTM ALTD)
    (LETS  ((CDTM (// FLTM 2)))
     (LIST (LIST 0 0) (LIST CDTM ALTD) (LIST FLTM 0))))

(DEFUN US:FLIGHT-PATTERN3 (FLTM) (LIST (LIST 0 0) (LIST FLTM 0)))

(DEFUN US:MAPPING-PATTERN-ON-ROUTE  (ROU SPED)
    (LETS  ((SET1 (@FRAME-FIND 'DEPARTURE-AIRPORT (NTH 0 ROU)))
            (SET2 (@FRAME-FIND 'ARRIVAL-AIRPORT (NTH 2 ROU)))
            (SET3 (@FGET 'ROUTE-LIST 'INSTANCE '¥LINK))
             (ROUT (CAR (US:INTERSECTION
                          (US:INTERSECTION SET1 SET3)
                          (US:INTERSECTION SET2 SET3))))
            (FIXS (@FGET ROUT 'PASSING-FIX '¥VALUE '(0)))
             (SGTM (DO
                       ((WORK1 FIXS (CDR WORK1))
                          (WORK2 0
                             (LETS
                                ((SEGM (US:CUTOUT-SEGMENT WORK1))
                                   (FLTM
                                      (US:SEGMENT-FLIGHT-TIME SEGM
                                        SPED)))
                                 (/+ WORK2 FLTM)))

                          (WORK3 NIL
                             (LETS
                                ((SEGM (US:CUTOUT-SEGMENT WORK1))
                                   (FLTM
                                      (US:SEGMENT-FLIGHT-TIME SEGM
                                        SPED))

                                   (FXTM
                                     (US:FIX-PASSING-TIME WORK2 FLTM)))
                                (PUSH (LIST SEGM FXTM) WORK3))))
                        ((EQ (LENGTH WORK1) 1) (REVERSE WORK3)))))
     (US:MAPPING-PATTERN-ON-SEGMENT SGTM)))

(DEFUN US:CUTOUT-SEGMENT (WORK1) (LIST (CAR WORK1) (CADR WORK1)))

(DEFUN US:SEGMENT-FLIGHT-TIME  (SEGM SPED)
    (LETS  ( (DIST (@FGET (CAR
                             (US:UNION
                                (US:INTERSECTION
                                  (@FRAME-FIND 'BEGIN-FIX (CAR SEGM))
                                  (@FRAME-FIND 'END-FIX (CADR SEGM)))

                                (US:INTERSECTION
                                  (@FRAME-FIND 'BEGIN-FIX (CADR SEGM))
                                  (@FRAME-FIND 'END-FIX (CAR SEGM)))))
                           'DISTANCE
                           '¥VALUE
                           '(0))))
     (FIX (/+¥ (*¥ 60.0 (//¥ (FLOAT DIST) (FLOAT SPED))) 0.5))))

(DEFUN US:FIX-PASSING-TIME (WORK2 FLTM) (LIST WORK2 (/+ WORK2 FLTM)))

(DEFUN US:MAPPING-PATTERN-ON-SEGMENT  (SGTM)
    (LETS  ((SEGTAC (US:CALCULATE-TIME-ALTITUDE-COORDINATE SGTM)))
     (US:FINAL-ARRANGEMENT-ON-EACH-SEGMENT SEGTAC)))

(DEFUN US:CALCULATE-TIME-ALTITUDE-COORDINATE  (SGTM)
    (MAPCAR  SGTM
       (FUNCTION
        (LAMBDA  (X)
         (APPENDX
                 (US:TIME-ALTITUDE-COORDINATE-ON-SEGMENT
                  (CADR X)))))))
```

付図２—11　プロダクション型知識ベース(11)

```
(DEFUN US:TIME-ALTITUDE-COORDINATE-ON-SEGMENT  (TT)
    (DO((WORK1 FPATTERN (CDR WORK1))
        (WORK2NIL
               (COND((US:INTERSECTION-TEST (CAR WORK1) (CADR WORK1)
                        TT)
                      (PUSH
                        (US:CUTOUT-FROM-SEGMENT (CAR WORK1)
                          (CADR WORK1) TT)
                        WORK2))
                     (T WORK2))))
       ((EQ (LENGTH WORK1) 1) (REVERSE WORK2))))

(DEFUN US:INTERSECTION-TEST  (P1 P2 TT)
    (LETS  ((P1T (CAR P1)) (P2T (CAR P2)) (T1T (CAR TT))
            (T2T (CADR TT)))
     (COND ((NOT (OR (<= P2T T1T) (<= T2T P1T))) T) (T NIL))))

(DEFUN US:CUTOUT-FROM-SEGMENT  (P1 P2 TT)
    (LETS  ((P1T (CAR P1)) (P2T (CAR P2)) (P1A (CADR P1))
            (P2A (CADR P2)) (T1T (CAR TT)) (T2T (CADR TT))
             (C1T (COND ((<= T1T P1T) P1T)
                        ((< P1T T1T) T1T)
                        (T (FORMAT "C1T-CUTOUT ERROR"))))

             (C2T (COND ((<= P2T T2T) P2T)
                        ((< T2T P2T) T2T)
                        (T (FORMAT "C2T-CUTOUT ERROR"))))

             (C1A (/+¥ (FLOAT P1A)
                       (*¥
                         (//¥ (FLOAT (/- C1T P1T))
                            (FLOAT (/- P2T P1T)))
                         (FLOAT (/- P2A P1A)))))

             (C2A (/+¥ (FLOAT P1A)
                       (*¥
                         (//¥ (FLOAT (/- C2T P1T))
                            (FLOAT (/- P2T P1T)))
                         (FLOAT (/- P2A P1A))))))
     (LIST(LIST C1T (* (// (/+ (FIX (/+¥ C1A 0.5)) 5) 10) 10))
          (LIST C2T (* (// (/+ (FIX (/+¥ C2A 0.5)) 5) 10) 10)))))

(DEFUN US:FINAL-ARRANGEMENT-ON-EACH-SEGMENT  (SEGTAC)
    (LETS  ((ARR1 (US:ARRANGE1 SEGTAC))
            (BEGN (CADR NOW-RUNWAY-OCCUPANCY-FOR-TAKE-OFF)))
     (US:ARRANGE2 ARR1 BEGN)))

(DEFUN US:ARRANGE1  (SEGTAC)
    (MAPCAR  SEGTAC
       (FUNCTION
        (LAMBDA  (X)
         (COND((EQ (LENGTH (CDDR X)) 1)
               (LIST (NTH 0 X) (CAR (NTH 2 X)) (CADR (NTH 2 X))))
              (
               (EQ (LENGTH (CDDR X)) 2)
               (LIST(NTH 0 X)
                     (CAR (NTH 2 X))
                     (CAR (NTH 3 X))
                     (CADR (NTH 3 X))))
              (
               (EQ (LENGTH (CDDR X)) 3)
               (LIST(NTH 0 X)
                     (CAR (NTH 2 X))
                     (CAR (NTH 3 X))
                     (CAR (NTH 4 X))
                     (CADR (NTH 4 X))))
              (T (FORMAT "ARRANGE1 ERROR")))))))

(DEFUN US:ARRANGE2  (ARR1 BEGN)
    (MAPCAR  ARR1
       (FUNCTION
        (LAMBDA  (X)
         (CONS (CAR X) (US:TIME-CONVERSION (CDR X) BEGN))))))

(DEFUN US:TIME-CONVERSION  (Y BEGN)
    (COND((NULL Y) NIL)
          (T(CONS (LIST (US:MIN-TO-HM
                          (/+ (CAAR Y) (US:HM-TO-MIN BEGN)))
                        (CADAR Y))
                  (US:TIME-CONVERSION (CDR Y) BEGN)))))

(DEFUN US:RUNWAY-OCCUPANCY-FOR-LANDING?  (LAST-CLEARANCE)
    (LETS  ((CAND (US:OCCUPANCY-CANDIDATE-L))
            (BLKD (US:BLOCKED-INTERVALS-L LAST-CLEARANCE)))
     (SETQ  NOW-RUNWAY-OCCUPANCY-FOR-LANDING
            (COND((NULL LAST-CLEARANCE) CAND)
                 (T (US:POSSIBLE-OCCUPANCY CAND BLKD))))
     NOW-RUNWAY-OCCUPANCY-FOR-LANDING))

(DEFUN US:OCCUPANCY-CANDIDATE-L  NIL
    (LETS  ((TYP NOW-TYPE)
            (BGN (CAR (CAR
                        (LAST
                           (CAR
                             (LAST NOW-FLIGHT-ON-EACH-SEGMENT))))))
            (END (US:LANDING-RUNWAY-OCCUPANCY-END-TIME BGN TYP)))
     (LIST BGN END)))

(DEFUN US:LANDING-RUNWAY-OCCUPANCY-END-TIME  (BGN TYP)
    (US:MIN-TO-HM
       (/+(US:HM-TO-MIN BGN)
```

付図２—12　プロダクション型知識ベース(12)

```
          (US:LANDING-RUNWAY-OCCUPANCY-TIME TYP))))

(DEFUN US:LANDING-RUNWAY-OCCUPANCY-TIME  (TYP)
    (LETS  ((TRK (CADR TYP))
            (IND (US:STRING-TO-NUMBER (SUBSTRING TRK 4 5)))
            (MTX (@VALUE '@DATA 'LANDING-RUNWAY-TIME-MATRIX))
             (DNF (COND ((= IND 1) (CAR MTX))
                        ((= IND 2) (CADR MTX))
                        ((= IND 3) (CADDR MTX))
                        (T
                          (FORMAT
                             "LANDING-RUNWAY-OCCUPANCY-TIME ERROR")))))
     (US:TIME-DECISION-BY-DNF-&-RN DNF)))

(DEFUN US:BLOCKED-INTERVALS-L  (LAST-CLEARANCE)
    (LETS  ((APOR NOW-ARRIVAL-AIRPORT)
            (SET1 (DO ((LINK LAST-CLEARANCE (@VALUE LINK 'LAST))
                       (CHAIN NIL (PUSH LINK CHAIN)))
                      ((NULL LINK) CHAIN)))

            (SET2 (@FIND-IN-LEVEL
                      'ATC-CLEARANCE
                      (LIST (LIST 'DEPARTURE-AIRPORT APOR))))

            (SET3 (@FIND-IN-LEVEL
                      'ATC-CLEARANCE
                      (LIST (LIST 'ARRIVAL-AIRPORT APOR))))
           (SET4 (US:INTERSECTION SET1 SET2))
           (SET5 (US:INTERSECTION SET1 SET3))
            (DBLK (COND ((NULL SET4) NIL)
                        (T
                          (MAPCAR SET4
                             (FUNCTION
                                (LAMBDA (X)
                                   (@VALUE X
                                      'RUNWAY-OCCUPANCY-FOR-TAKE-OFF)))))))

            (ABLK (COND ((NULL SET5) NIL)
                        (T
                          (MAPCAR SET5
                             (FUNCTION
                                (LAMBDA (X)
                                   (@VALUE X
                                      'RUNWAY-OCCUPANCY-FOR-LANDING)))))))
           (BLKD (US:UNION DBLK ABLK)))
     (COND((NULL BLKD) NIL)
           (T(SORT BLKD
                   (FUNCTION
                    (LAMBDA (U V) (< (CAR U) (CAR V)))))))))

(DEFUN US:TAXIING-AFTER-LANDING?  NIL
    (LETS  ((TYP NOW-TYPE)
            (BGN (CADR NOW-RUNWAY-OCCUPANCY-FOR-LANDING))
            (END (US:LANDING-TAXI-END-TIME BGN TYP)))
     (SETQ NOW-TAXIING-AFTER-LANDING (LIST BGN END))
     NOW-TAXIING-AFTER-LANDING))

(DEFUN US:LANDING-TAXI-END-TIME  (BGN TYP)
    (US:MIN-TO-HM
       (/+ (US:HM-TO-MIN BGN) (US:LANDING-TAXI-TIME TYP))))

(DEFUN US:LANDING-TAXI-TIME  (TYP)
    (LETS  ((TRK (CADR TYP))
            (IND (US:STRING-TO-NUMBER (SUBSTRING TRK 4 5)))
            (MTX (@VALUE '@DATA 'LANDING-TAXI-TIME-MATRIX))
             (DNF (COND ((= IND 1) (CAR MTX))
                        ((= IND 2) (CADR MTX))
                        ((= IND 3) (CADDR MTX))
                        (T (FORMAT "LANDIND-TAXI-TIME ERROR")))))
     (US:TIME-DECISION-BY-DNF-&-RN DNF)))

(DEFUN US:ORDER-PLUS-ONE  (LAST-CLEARANCE)
    (/+ (@VALUE LAST-CLEARANCE 'ORDER) 1))

(DEFUN US:FLIGHT-LEVELS  (ORD)
    (LETS  ((TR (US:TYPE-RANK2 ORD)) (RR (US:ROUTE-RANK2 ORD))
            (RD (US:ROUTE-DIRECTION2 ORD))
            (FLS (@VALUE '@DATA 'FLIGHT-LEVELS)))
     (US:SELECT-POSSIBLE-LEVELS TR RR RD FLS)))

(DEFUN US:TYPE-RANK2 (ORD) (NTH 1 (US:TYPE? ORD)))

(DEFUN US:ROUTE-RANK2 (ORD) (NTH 3 (US:ROUTE? ORD)))

(DEFUN US:ROUTE-DIRECTION2  (ORD)
    (LETS  ((FSCH (CAR (@FRAME-FIND 'ORDER ORD)))
            (RDEP (@FGET FSCH 'DEPARTURE-AIRPORT '¥VALUE '(0)))
            (RARR (@FGET FSCH 'ARRIVAL-AIRPORT '¥VALUE '(0)))
            (SET1 (@FRAME-FIND 'DEPARTURE-AIRPORT RDEP))
            (SET2 (@FRAME-FIND 'ARRIVAL-AIRPORT RARR))
            (SET3 (@FGET 'ROUTE-LIST 'INSTANCE '¥LINK))
             (ROUT (CAR
                       (US:INTERSECTION (US:INTERSECTION SET1 SET3)
                         (US:INTERSECTION SET2 SET3)))))
     (@FGET ROUT 'DIRECTION '¥VALUE '(0))))

(DEFUN US:ALTITUDE-IN-DEFAULT?  (FL)
    (SETQ NOW-ALTITUDE FL)
    (US:CONFIRM-ALTITUDE))

(DEFUN US:CONFLICT-CHECK  (FRESH-CLEARANCE)
```

付図２—13　プロダクション型知識ベース(13)

```
    (DO((WORK (@VALUE FRESH-CLEARANCE 'LAST) (@VALUE WORK 'LAST)))
        ((NULL WORK) NIL)
        (COND((US:CHECK-CLR-TO-CLR FRESH-CLEARANCE WORK) (EXIT T)))))

(DEFUN US:CHECK-CLR-TO-CLR  (FCLR LCLR)
    (LETS  ((FALT (@VALUE FCLR 'ALTITUDE))
            (LALT (@VALUE LCLR 'ALTITUDE)))
     (COND((OR (EQ FALT 0) (EQ LALT 0) (NEQ FALT LALT)) NIL)
           (T(LETS  ( (FSEG
                        (US:REMOVE-NIL (US:SEGMENT-SET FCLR FALT)))

                      (LSEG
                        (US:REMOVE-NIL (US:SEGMENT-SET LCLR LALT)))
                     (FFIX (US:REMOVE-NIL (US:FIX-SET FCLR FALT)))
                     (LFIX (US:REMOVE-NIL (US:FIX-SET LCLR LALT))))
                (COND((AND (NULL (US:SEG-INTERSECTION FSEG LSEG))
                           (NULL (US:FIX-INTERSECTION FFIX LFIX)))
                      NIL)
                     ((US:CHECK-SEG-TO-SEG FSEG FALT LSEG LALT) T)
                     ((US:CHECK-FIX-TO-FIX FFIX FALT LFIX LALT) T)
                     (T NIL)))))))

(DEFUN US:SEGMENT-SET  (CLR ALT)
    (MAPCAR  (@VALUE CLR 'FLIGHT-ON-EACH-SEGMENT)
       (FUNCTION
        (LAMBDA  (X)
         (COND((SOME  (CDR X)
                      (FUNCTION (LAMBDA (Y) (EQ (CADR Y) ALT))))
               X)
              (T NIL))))))

(DEFUN US:FIX-SET  (CLR ALT)
    (MAPCAR  (@VALUE CLR 'FLIGHT-ON-EACH-SEGMENT)
       (FUNCTION
        (LAMBDA  (X)
         (COND((EQ (CADADR X) ALT) (LIST (CAAR X) (CAADR X)))
               (T NIL))))))

(DEFUN US:REMOVE-NIL  (LST)
    (DO((WORK1 LST (CDR WORK1))
        (WORK2NIL
              (COND((NULL (CAR WORK1)) WORK2)
                    (T (PUSH (CAR WORK1) WORK2)))))
       ((NULL WORK1) (REVERSE WORK2))))

(DEFUN US:SEG-INTERSECTION  (FSEG LSEG)
    (US:INTERSECTION
       (MAPCAR FSEG (FUNCTION (LAMBDA (F) (CAR F))))
       (MAPCAR LSEG (FUNCTION (LAMBDA (L) (CAR L))))))

(DEFUN US:FIX-INTERSECTION  (FFIX LFIX)
    (US:INTERSECTION
       (MAPCAR FFIX (FUNCTION (LAMBDA (F) (CAR F))))
       (MAPCAR LFIX (FUNCTION (LAMBDA (L) (CAR L))))))

(DEFUN US:CHECK-SEG-TO-SEG  (FSEG FALT LSEG LALT)
    (LETS  ((F1 (US:FIRST-FALT-POINT FSEG FALT))
            (F2 (US:LAST-FALT-POINT FSEG FALT))
            (L1 (US:FIRST-LALT-POINT FSEG LSEG LALT))
            (L2 (US:LAST-LALT-POINT FSEG LSEG LALT)))
     (US:CHECK-DISTANCE-MINIMUM F1 F2 L1 L2)))

(DEFUN US:FIRST-FALT-POINT  (FSEG FALT)
    (LETS  ((FST (CAR FSEG)) (DST (US:SEGMENT-DISTANCE (CAR FST)))
            (BGN (CAR (SECOND FST))) (END (CAR (CAR (LAST FST))))
             (F1T (CAR (CAR
                          (SOME (CDR FST)
                             (FUNCTION
                               (LAMBDA (X) (EQ (CADR X) FALT)))))))

             (F1D (COND ((EQ BGN END) 0)
                        (T
                          (*¥
                             (//¥ (FLOAT DST)
                                (FLOAT
                                   (/- (US:HM-TO-MIN END)
                                     (US:HM-TO-MIN BGN))))

                                (FLOAT
                                   (/- (US:HM-TO-MIN F1T)
                                     (US:HM-TO-MIN BGN)))))))))
     (LIST F1T (FIX (/+¥ F1D 0.5)))))

(DEFUN US:SEGMENT-DISTANCE  (SEG)
    (@FGET(CAR (US:UNION
                  (US:INTERSECTION
                     (@FRAME-FIND 'BEGIN-FIX (CAR SEG))
                     (@FRAME-FIND 'END-FIX (CADR SEG)))
                  (US:INTERSECTION
                     (@FRAME-FIND 'BEGIN-FIX (CADR SEG))
                     (@FRAME-FIND 'END-FIX (CAR SEG)))))
          'DISTANCE
          '¥VALUE
          '(0)))

(DEFUN US:LAST-FALT-POINT  (FSEG FALT)
    (LETS  ((LST (CAR (LAST FSEG)))
            (DST (US:SEGMENT-DISTANCE (CAR LST)))
            (BGN (CAR (SECOND LST))) (END (CAR (CAR (LAST LST))))
             (F2T (CAR (CAR
                          (SOME (REVERSE (CDR LST))
```

付図２—14　プロダクション型知識ベース(14)

```
                                (FUNCTION
                                  (LAMBDA (X) (EQ (CADR X) FALT)))))))

              (F2D (COND ((EQ BGN END) 0)
                         (T
                           (*¥
                              (//¥ (FLOAT DST)
                                 (FLOAT
                                    (/- (US:HM-TO-MIN END)
                                      (US:HM-TO-MIN BGN))))

                              (FLOAT
                                 (/- (US:HM-TO-MIN F2T)
                                   (US:HM-TO-MIN BGN)))))))
            (SUM (US:DISTANCE-SUM-EXCEPT-LAST-SEGMENT FSEG)))
     (LIST F2T (/+ SUM (FIX (/+¥ F2D 0.5))))))

(DEFUN US:DISTANCE-SUM-EXCEPT-LAST-SEGMENT  (FSEG)
    (LETS  ( (WSEG (CDR (REVERSE
                          (MAPCAR FSEG
                            (FUNCTION (LAMBDA (X) (CAR X))))))))
     (US:DISTANCE-SUM WSEG)))

(DEFUN US:DISTANCE-SUM  (WSEG)
    (COND((NULL WSEG) 0)
          (T(/+ (US:SEGMENT-DISTANCE (CAR WSEG))
                (US:DISTANCE-SUM (CDR WSEG))))))

(DEFUN US:FIRST-LALT-POINT  (FSEG LSEG LALT)
    (LETS  ((CSEG (US:CUTOUT-COMMON-PART FSEG LSEG)))
     (COND((US:CONNECTEDNESS-CHECK CSEG)
           (FORMAT "CONNECTEDNESS-CHECK OUT"))
          (T (US:CALCULATE-L1 FSEG CSEG LALT)))))

(DEFUN US:CUTOUT-COMMON-PART  (FSEG LSEG)
    (US:REMOVE-NIL
       (LETS  ((COMM (US:SEG-INTERSECTION FSEG LSEG)))
        (MAPCAR  LSEG
           (FUNCTION
            (LAMBDA  (X)
             (COND ((MEMBER (CAR X) COMM) X) (T NIL))))))))

(DEFUN US:CONNECTEDNESS-CHECK  (CSEG)
    (DO((WORK CSEG (CDR WORK)))
       ((EQ (LENGTH WORK) 1) NIL)
       (COND((NOT (EQUAL (CADAR (FIRST WORK)) (CAAR (SECOND WORK))))
             (EXIT T)))))

(DEFUN US:CALCULATE-L1  (FSEG CSEG LALT)
    (LETS  ((FST (CAR CSEG))
            (COR (DO ((WORK1 FSEG (CDR WORK1))
                      (WORK2 0
                         (/+ WORK2
                            (US:SEGMENT-DISTANCE
                              (CAR (CAR WORK1))))))
                     ((EQUAL (CAR (CAR WORK1)) (CAR FST)) WORK2)))
            (DST (US:SEGMENT-DISTANCE (CAR FST)))
            (BGN (CAR (SECOND FST))) (END (CAR (CAR (LAST FST))))
            (L1T (CAR (CAR
                        (SOME (CDR FST)
                           (FUNCTION
                             (LAMBDA (X) (EQ (CADR X) LALT)))))))

            (L1D (COND ((EQ BGN END) 0)
                       (T
                         (*¥
                            (//¥ (FLOAT DST)
                               (FLOAT
                                  (/- (US:HM-TO-MIN END)
                                    (US:HM-TO-MIN BGN))))

                            (FLOAT
                               (/- (US:HM-TO-MIN L1T)
                                 (US:HM-TO-MIN BGN)))))))))
     (LIST L1T (/+ COR (FIX (/+¥ L1D 0.5))))))

(DEFUN US:LAST-LALT-POINT  (FSEG CSEG LALT)
    (LETS  ((CSEG (US:CUTOUT-COMMON-PART FSEG LSEG)))
     (COND((US:CONNECTEDNESS-CHECK CSEG)
           (FORMAT "CONNECTEDNESS-CHECK OUT"))
          (T (US:CALCULATE-L2 FSEG CSEG LALT)))))

(DEFUN US:CALCULATE-L2  (FSEG CSEG LALT)
    (LETS  ((FST (CAR CSEG))
            (COR (DO ((WORK1 FSEG (CDR WORK1))
                      (WORK2 0
                         (/+ WORK2
                            (US:SEGMENT-DISTANCE
                              (CAR (CAR WORK1))))))
                     ((EQUAL (CAR (CAR WORK1)) (CAR FST)) WORK2)))
            (LST (CAR (LAST CSEG)))
            (DST (US:SEGMENT-DISTANCE (CAR LST)))
            (BGN (CAR (SECOND LST))) (END (CAR (CAR (LAST LST))))
            (L2T (CAR (CAR
                        (SOME (REVERSE (CDR LST))
                           (FUNCTION
                             (LAMBDA (X) (EQ (CADR X) LALT)))))))

            (L2D (COND ((EQ BGN END) 0)
                       (T
                         (*¥
```

付図２—15　プロダクション型知識ベース(15)

```
                                (//¥ (FLOAT DST)
                                   (FLOAT
                                      (/- (US:HM-TO-MIN END)
                                        (US:HM-TO-MIN BGN))))

                                (FLOAT
                                   (/- (US:HM-TO-MIN L2T)
                                     (US:HM-TO-MIN BGN)))))))
            (SUM (US:DISTANCE-SUM-EXCEPT-LAST-SEGMENT CSEG)))
     (LIST L2T (/+ COR SUM (FIX (/+¥ L2D 0.5))))))

(DEFUN US:CHECK-DISTANCE-MINIMUM  (F1 F2 L1 L2)
    (LETS  ((F1T (CAR F1)) (F2T (CAR F2)) (L1T (CAR L1))
            (L2T (CAR L2)))
     (COND((OR (< L2T F1T) (< F2T L1T)) NIL)
           (T (US:CHECK-ON-COMMON-TIME-INTERVAL F1 F2 L1 L2)))))

(DEFUN US:CHECK-ON-COMMON-TIME-INTERVAL  (F1 F2 L1 L2)
    (LETS  ((F1T (CAR F1)) (F2T (CAR F2)) (L1T (CAR L1))
            (L2T (CAR L2))
             (C1T (COND ((<= L1T F1T) F1T)
                        ((< F1T L1T) L1T)
                        (T (FORMAT "COMMON-TIME-INTERVAL ERROR"))))

             (C2T (COND ((<= F2T L2T) F2T)
                        ((< L2T F2T) L2T)
                        (T (FORMAT "COMMON-TIME-INTERVAL ERROR"))))
            (FC1D (US:CALCULATE-POSITION F1 F2 C1T))
            (FC2D (US:CALCULATE-POSITION F1 F2 C2T))
            (LC1D (US:CALCULATE-POSITION L1 L2 C1T))
            (LC2D (US:CALCULATE-POSITION L1 L2 C2T)))
     (US:CONFLICT-CHECK1 FC1D FC2D LC1D LC2D)))

(DEFUN US:CALCULATE-POSITION  (BGN END CTM)
    (LETS  ((BGNT (CAR BGN)) (ENDT (CAR END)) (BGND (CADR BGN))
            (ENDD (CADR END))
             (CDST (COND ((EQ BGNT ENDT) 0)
                         (T
                            (*¥
                               (//¥ (FLOAT (/- ENDD BGND))
                                  (FLOAT
                                     (/- (US:HM-TO-MIN ENDT)
                                       (US:HM-TO-MIN BGNT))))

                               (FLOAT
                                  (/- (US:HM-TO-MIN CTM)
                                    (US:HM-TO-MIN BGNT))))))))
     (FIX (/+¥ CDST 0.5))))

(DEFUN US:CONFLICT-CHECK1  (FC1D FC2D LC1D LC2D)
    (COND((OR (EQ FC1D LC1D) (EQ FC2D LC2D)) T)
          (T (US:CONFLICT-CHECK2 FC1D FC2D LC1D LC2D))))

(DEFUN US:CONFLICT-CHECK2  (FC1D FC2D LC1D LC2D)
    (COND((OR (AND (/0< (/- FC1D LC1D)) (/0> (/- FC2D LC2D)))
              (AND (/0> (/- FC1D LC1D)) (/0< (/- FC2D LC2D))))
          T)
          (T (US:CONFLICT-CHECK3 FC1D FC2D LC1D LC2D))))

(DEFUN US:CONFLICT-CHECK3  (FC1D FC2D LC1D LC2D)
    (LETS  ((SEPA (@VALUE '@DATA 'SEPARATION-STANDARD)))
     (COND((< (MIN (ABS (/- FC1D LC1D)) (ABS (/- FC2D LC2D))) SEPA)
           T)
           (T NIL))))

(DEFUN US:CHECK-FIX-TO-FIX  (FFIX FALT LFIX LALT)
    (COND((SOME  FFIX
                 (FUNCTION
                  (LAMBDA (X) (US:FIND-CONFLICTION-ON-FIX X LFIX))))
          T)
          (T NIL)))

(DEFUN US:FIND-CONFLICTION-ON-FIX  (X LFIX)
    (COND((SOME  LFIX
                 (FUNCTION
                  (LAMBDA  (Y)
                   (AND(EQUAL (CAR X) (CAR Y))
                       (EQUAL (CADR X) (CADR Y))))))
          T)
          (T NIL)))

(DEFUN US:SOLUTION-OUTPUT  (END-CLEARANCE)
    (LETS  ((ENDFLT (@VALUE '@DATA 'END-FLIGHT))
            (SLTION (DO
                        ((X END-CLEARANCE (@VALUE X 'LAST))
                          (Y NIL (PUSH X Y)))
                        ((NULL X) Y))))
     (FORMAT
       "WE FOUND A SOLUTION IN WHICH /C FLIGHTS DO NOT CONFLICT EACH OTHER /N"
      ENDFLT)
     (FORMAT "/C /N" SLTION)))

*** 設計終了 ***
```

付図２—16　プロダクション型知識ベース(16)

```
(@DEFRAME
   'AIR-TRAFFIC-DATA
   '(CLASS (¥VALUE (GENERIC)))
   '(SUBC (¥LINK (AIRPORT-LIST)
                 (AIRLINE-LIST)
                 (TYPE-LIST)
                 (FLIGHT-SCHEDULE-LIST)
                 (IDLE-LINE-LIST)
                 (FIX-LIST)
                 (SEGMENT-LIST)
                 (ROUTE-LIST))))

(@DEFRAME
   'AIRPORT
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (AIRPORT-LIST)))
   '(NUMBER)
   '(ABBREV)
   '(NAME))

(@DEFRAME
   'AIRPORT-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
   'AIRLINE
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(NUMBER)
   '(ABBREV)
   '(NAME))

(@DEFRAME
   'AIRLINE-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
   'TYPE
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (TYPE-LIST)))
   '(NUMBER)
   '(ABBREV)
   '(CODE)
   '(NAME))

(@DEFRAME
   'TYPE-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
   'FLIGHT-SCHEDULE
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (FLIGHT-SCHEDULE-LIST)))
   '(NUMBER)
   '(DEPARTURE-AIRPORT)
   '(ARRIVAL-AIRPORT)
   '(AIRLINE)
   '(FLIGHT-NUMBER)
   '(TYPE)
   '(DEPARTURE-TIME)
   '(ARRIVAL-TIME))

(@DEFRAME
   'FLIGHT-SCHEDULE-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
   'IDLE-LINE
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (IDLE-LINE-LIST)))
   '(NUMBER)
   '(DEPARTURE-AIRPORT)
   '(ARRIVAL-AIRPORT))

(@DEFRAME
   'IDLE-LINE-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
   'FIX
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (FIX-LIST)))
   '(NUMBER)
   '(ABBREV)
   '(NAME))

(@DEFRAME
   'FIX-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
   'SEGMENT
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(AKO (¥LINK (SEGMENT-LIST)))
   '(NUMBER)
```

付図３－１　フレーム構成(1)

```
    '(BEGIN-FIX)
    '(END-FIX)
    '(DISTANCE))

(@DEFRAME
    'SEGMENT-LIST
    '(CLASS (¥VALUE (GENERIC)))
    '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(@DEFRAME
    'ROUTE
    '(CLASS (¥VALUE (INDIVIDUAL)))
    '(AKO (¥LINK (ROUTE-LIST)))
    '(NUMBER)
    '(DEPARTURE-AIRPORT)
    '(ARRIVAL-AIRPORT)
    '(DIRECTION)
    '(PASSING-FIX))

(@DEFRAME
    'ROUTE-LIST
    '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA))))

(COMMENT (USER FUNCTION))
```

付図３－２　フレーム構成(2)

```
(@DEFRAME
   'AIR-TRAFFIC-DATA
   '(CLASS (¥VALUE (GENERIC)))
   '(SUBC (¥LINK (AIRPORT-LIST)
                 (AIRLINE-LIST)
                 (TYPE-LIST)
                 (FLIGHT-SCHEDULE-LIST)
                 (IDLE-LINE-LIST)
                 (FIX-LIST)
                 (SEGMENT-LIST)
                 (ROUTE-LIST))))

(@DEFRAME
   'AIRLINE-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA)))
   '(INSTANCE
       (¥LINK (AIRLINE0001)
              (AIRLINE0002)
              (AIRLINE0003)
              (AIRLINE0004)
              (AIRLINE0005)
              (AIRLINE0006))))

(@DEFRAME
   'AIRLINE0001
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(NUMBER (¥VALUE (1)))
   '(ABBREV (¥VALUE (JAL)))
   '(NAME (¥VALUE (JAPANAIR))))

(@DEFRAME
   'AIRLINE0002
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(NUMBER (¥VALUE (2)))
   '(ABBREV (¥VALUE (ANA)))
   '(NAME (¥VALUE (ALL-NIPPON))))

(@DEFRAME
   'AIRLINE0003
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(NUMBER (¥VALUE (3)))
   '(ABBREV (¥VALUE (JAS)))
   '(NAME (¥VALUE (AIR-SYSTEM))))

(@DEFRAME
   'AIRLINE0004
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(NUMBER (¥VALUE (4)))
   '(ABBREV (¥VALUE (SWL)))
   '(NAME (¥VALUE (NANSEI))))

(@DEFRAME
   'AIRLINE0005
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(NUMBER (¥VALUE (5)))
   '(ABBREV (¥VALUE (ANK)))
   '(NAME (¥VALUE (ANK-AIR))))

(@DEFRAME
   'AIRLINE0006
   '(AKO (¥LINK (AIRLINE-LIST)))
   '(CLASS (¥VALUE (INDIVIDUAL)))
   '(NUMBER (¥VALUE (6)))
   '(ABBREV (¥VALUE (JAC)))
   '(NAME (¥VALUE (COMMUTER))))

(@DEFRAME
   'AIRPORT-LIST
   '(CLASS (¥VALUE (GENERIC)))
   '(SUPERC (¥LINK (AIR-TRAFFIC-DATA)))
   '(INSTANCE
       (¥LINK (AIRPORT0001)
              (AIRPORT0002)
              (AIRPORT0003)
              (AIRPORT0004)
              (AIRPORT0005)
              (AIRPORT0006)
              (AIRPORT0007)
              (AIRPORT0008)
              (AIRPORT0009)
              (AIRPORT0010)
              (AIRPORT0011)
              (AIRPORT0012)
              (AIRPORT0013)
              (AIRPORT0014)
              (AIRPORT0015)
              (AIRPORT0016)
              (AIRPORT0017)
              (AIRPORT0018)
              (AIRPORT0019)
              (AIRPORT0020)
              (AIRPORT0021)
              (AIRPORT0022)
              (AIRPORT0023)
              (AIRPORT0024)
```

付図4－1　データ入力後のフレーム(部分)(1)

```
                (AIRPORT0025)
                (AIRPORT0026)
                (AIRPORT0027)
                (AIRPORT0028)
                (AIRPORT0029)
                (AIRPORT0030)
                (AIRPORT0031)
                (AIRPORT0032)
                (AIRPORT0033)
                (AIRPORT0034)
                (AIRPORT0035)
                (AIRPORT0036)
                (AIRPORT0037)
                (AIRPORT0038)
                (AIRPORT0039)
                (AIRPORT0040)
                (AIRPORT0041)
                (AIRPORT0042)
                (AIRPORT0043)
                (AIRPORT0044)
                (AIRPORT0045)
                (AIRPORT0046)
                (AIRPORT0047)
                (AIRPORT0048)
                (AIRPORT0049)
                (AIRPORT0050)
                (AIRPORT0051)
                (AIRPORT0052)
                (AIRPORT0053)
                (AIRPORT0054)
                (AIRPORT0055)
                (AIRPORT0056)
                (AIRPORT0057)
                (AIRPORT0058)
                (AIRPORT0059)
                (AIRPORT0060)
                (AIRPORT0061)
                (AIRPORT0062)
                (AIRPORT0063)
                (AIRPORT0064)
                (AIRPORT0065)
                (AIRPORT0066)
                (AIRPORT0067)
                (AIRPORT0068))))

(@DEFFRAME
    'AIRPORT0001
    '(AKO (¥LINK (AIRPORT-LIST)))
    '(CLASS (¥VALUE (INDIVIDUAL)))
    '(NUMBER (¥VALUE (1)))
    '(ABBREV (¥VALUE (JCR)))
    '(NAME (¥VALUE (REBUN))))

(@DEFFRAME
    'AIRPORT0002
    '(AKO (¥LINK (AIRPORT-LIST)))
    '(CLASS (¥VALUE (INDIVIDUAL)))
    '(NUMBER (¥VALUE (2)))
    '(ABBREV (¥VALUE (JER)))
    '(NAME (¥VALUE (RISHIRI))))

(@DEFFRAME
    'AIRPORT0003
    '(AKO (¥LINK (AIRPORT-LIST)))
    '(CLASS (¥VALUE (INDIVIDUAL)))
    '(NUMBER (¥VALUE (3)))
    '(ABBREV (¥VALUE (JCW)))
    '(NAME (¥VALUE (WAKKANAI))))

(@DEFFRAME
    'AIRPORT0004
    '(AKO (¥LINK (AIRPORT-LIST)))
    '(CLASS (¥VALUE (INDIVIDUAL)))
    '(NUMBER (¥VALUE (4)))
    '(ABBREV (¥VALUE (JEB)))
    '(NAME (¥VALUE (MONBETSU))))

(@DEFFRAME
    'AIRPORT0005
    '(AKO (¥LINK (AIRPORT-LIST)))
    '(CLASS (¥VALUE (INDIVIDUAL)))
    '(NUMBER (¥VALUE (5)))
    '(ABBREV (¥VALUE (JCM)))
    '(NAME (¥VALUE (MENANBETSU))))
          |
          |
          |
          |
          |
          |
          |
          |
          |
```

付図４－２　データ入力後のフレーム(部分)(2)